CS 353 DATABASE SYSTEMS
SPRING 2021

PROJECT DESIGN

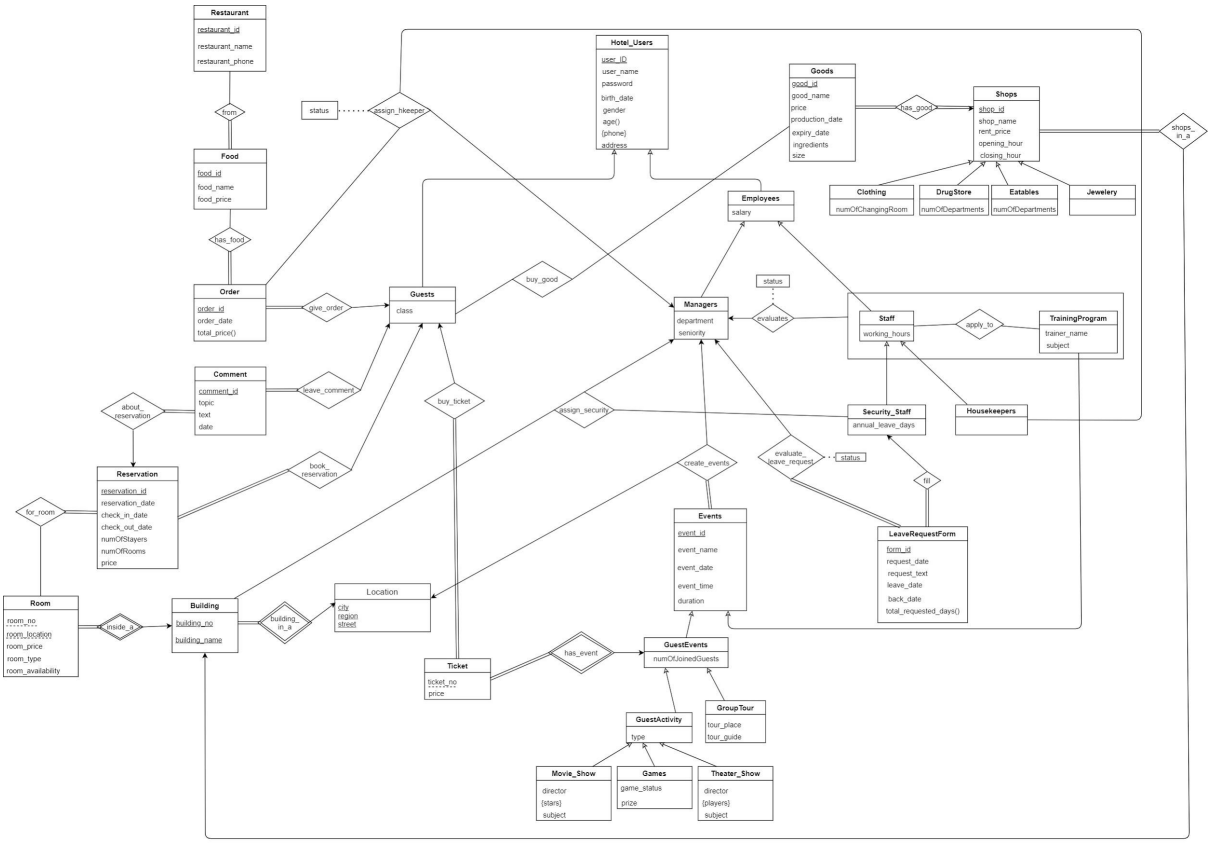
Hotel Database Management System

Group 37
Abdullah Ayberk Görgün - 21201986
Cemal Gündüz - 21703004
Veli Can Mert - 21602394
Hande Sena Yılmaz - 21703465

1. Introduction

In this design report of the project, we will specify the designing details of Hotel Database Management System web server which includes ER diagram and Relations to represent the database management of the server and sample User-Interface designs for front-end of the system and their corresponding SQL statements for back-end of the system. Finally, we will mention the implementation plan of the web server system including what are planned to use for dbm systems and development technologies.

2. Revised ER Diagram



3. Relations

3.1 Entities

3.1.1 Hotel_Users

Relational Model

Hotel_Users(<u>user_ID</u>, user_name, password, birth_date, gender, address)

Functional Dependencies

user_ID → user_name password birth_date gender address

Candidate Keys

{(user_ID)}

Primary Key

(user_ID)

Table Definition

```
CREATE TABLE Hotel_User(
user_ID          INT not null auto_increment,
user_name        VARCHAR(20) not null,
password         VARCHAR(20) not null,
birth_date       DATE,
gender           VARCHAR(6),
address          VARCHAR(100),
PRIMARY KEY (user_ID)
) ENGINE = InnoDB;
```

3.1.2 Reservation

Relational Model

Reservation(reservation_id, reservation_date, check_in_date, check_out_date,numOfStayers, numOfRooms, price, user_ID)

foreign key: user_ID references Guests

Functional Dependencies

reservation_id → reservation_date check_in_date check_out_date numOfStayers numOfRooms price user_ID

Candidate Keys

{(reservation_id)}

Primary Key

(reservation_id)

Table Definition

```
CREATE TABLE Reservation(
reservation_id      INT not null auto_increment,
reservation_date    DATE not null
check_in_date       DATE not null,
check_out_date      DATE not null,
numOfStayers        INT not null,
numOfRooms          INT not null,
price               INT not null,
user_ID             INT not null,
PRIMARY KEY (reservation_id),
FOREIGN KEY (user_ID) REFERENCES Guests
) ENGINE = InnoDB;
```

3.1.3 Goods

Relational Model

Goods(good_id, shop_id, good_name, price, production_date, expiry_date, ingredients, size)

foreign key: shop_id references Shops

Functional Dependencies

good_id shop_id → good_name price production_date expiry_date ingredients size

Candidate Keys

{(good_id,shop_id)}

Primary Key

(good_id)

Table Definition

```
CREATE TABLE Goods(
good_id          INT not null auto_increment,
good_name        VARCHAR(20) not null,
price            INT not null,
production_date  DATE,
expiry_date      DATE,
ingredients      VARCHAR(200),
size             VARCHAR (100),
PRIMARY KEY (good_id, shop_id),
FOREIGN KEY (shop_id) REFERENCES Shops
) ENGINE = InnoDB;
```

3.1.4 Shops

Relational Model

Shops(shop_id, shop_name, rent_price, opening_hour, closing_hour)

Functional Dependencies

shop_id → shop_name rent_price opening_hour closing_hour

Candidate Keys

{(shop_id)}

Primary Key

(shop_id)

Table Definition

```
CREATE TABLE Shops(
shop_id          INT not null auto_increment,
shop_name        VARCHAR(20) not null,
rent_price       INT not null,
opening_hour     CHAR(5),
closing_hour     CHAR(5),
PRIMARY KEY (shop_id)
) ENGINE = InnoDB;
```

3.1.5 Room

Relational Model

Room(<u>room_no, room_location, building_no, building_name</u>, room_price, room_type, room_availability)

foreign key: (building_no, building_name) references Building

Functional Dependencies

room_no room_location building_no building_name → room_price room_price room_type room_availability

Candidate Keys

{(room_no, room_location, building_no, building_name)}

Primary Key

(room_no, room_location, building_no, building_name)

Table Definition

```
CREATE TABLE Room(
room_no            INT not null auto_increment,
room_location      VARCHAR(20) not null,
building_no        INT not_null,
building_name      VARCHAR(20) not_null,
room_price         INT not null,
room_type          VARCHAR(20),
room_availability  VARCHAR(20),
PRIMARY KEY (room_no, room_location, building_no, building_name),
FOREIGN KEY (building_no, building_name) REFERENCES Building
) ENGINE = InnoDB;
```

3.1.6 Building

Relational Model

Building(<u>building_no, building_name,city, region, street</u>)

foreign key: (city, region, street) references Location

Functional Dependencies

None

Candidate Keys

{(building_no, building_name, city, region, street)}

Primary Key

(building_no, building_name, city, region, street)

Table Definition

CREATE TABLE Building(

building_no **INT** not null auto_increment,

building_name **VARCHAR(20)** not null,

city **VARCHAR(20)** not null,

region **VARCHAR(20)** not null,

street **VARCHAR(20)** not null,

PRIMARY KEY (building_no, building_name, city, region, street),

FOREIGN KEY (city, region, street) REFERENCES Location

) ENGINE = InnoDB;

3.1.7 Location

Relational Model

Location(<u>city, region, street</u>)

Functional Dependencies

None

Candidate Keys

{(city,region,street)}

Primary Key

(city, region, street)

Table Definition

CREATE TABLE Location(

city **VARCHAR(20)** not null,

region **VARCHAR(20)** not null,

street **VARCHAR(20)** not null,

PRIMARY KEY (city, region, street)

) ENGINE = InnoDB;

3.1.8 Comment

Relational Model

Comment(comment_id, topic, text, date, reservation_id, user_ID)

foreign key: reservation_id references Reservation

foreign key: user_ID references Guests

Functional Dependencies

comment_id → topic text date reservation_id user_ID

Candidate Keys

{(comment_id)}

Primary Key

(comment_id)

Table Definition

```
CREATE TABLE Comment(
comment_id      INT not null auto_increment,
topic           VARCHAR(15),
text            VARCHAR(300),
date            DATE,
reservation_id  INT not null,
user_ID         INT not null,
PRIMARY KEY (comment_id),
FOREIGN KEY (reservation_id) REFERENCES Reservation,
FOREIGN KEY (user_ID) REFERENCES Guests
```

) ENGINE = InnoDB;

3.1.9 Restaurant

Relational Model

Restaurant(restaurant_id,restaurant_name,restaurant_phone)

Functional Dependencies

restaurant_id → restaurant_name restaurant_phone

Candidate Keys

{(restaurant_id)}

Primary Key

(restaurant_id)

Table Definition

```
CREATE TABLE Restaurant(
restaurant_id        INT not null auto_increment,
restaurant_name      VARCHAR(20) not null,
restaurant_phone     CHAR(10),
PRIMARY KEY (restaurant_id)
) ENGINE = InnoDB;
```

3.1.10 Food

Relational Model

Food(food_id,food_name, food_price, restaurant_id)

foreign key: restaurant_id references Restaurant

Functional Dependencies

food_id → food_name food_price restaurant_id

Candidate Keys

{(food_id)}

Primary Key

(food_id)

Table Definition

```
CREATE TABLE Food(
food_id          INT not null auto_increment,
restaurant_id    INT not null,
food_name        VARCHAR(20) not null,
food_price       INT not null,
PRIMARY KEY (food_id),
FOREIGN KEY (restaurant_id) REFERENCES Restaurant
) ENGINE = InnoDB;
```

3.1.11 Order

Relational Model

Order(<u>order_id</u>, order_date, food_id, user_ID)

foreign key: food_id references Food

foreign key: user_ID references Guests

Functional Dependencies

order_id → order_date food_id user_ID

Candidate Keys

{(order_id)}

Primary Key

(order_id)

Table Definition

CREATE TABLE Order(

order_id **INT** not null auto_increment,

order_date **DATE** not null,

food_id **INT** not null,

user_ID **INT** not null,

PRIMARY KEY (order_id),

FOREIGN KEY (food_id) REFERENCES Food,

FOREIGN KEY (user_ID) REFERENCES Guests

) ENGINE = InnoDB;

3.1.12 Guests

Relational Model

Guests(user_ID, class)

foreign key: user_ID references Hotel_Users

Functional Dependencies

user_ID → class

Candidate Keys

{(user_ID)}

Primary Key

(user_ID)

Table Definition

CREATE TABLE Guests(

user_ID **INT** not null,

class **CHAR(1)** not null,

PRIMARY KEY (user_ID),

FOREIGN KEY (user_ID) REFERENCES Hotel_Users,

) ENGINE = InnoDB;

3.1.13 Events

Relational Model

Events(event_id, event_name, event_date, event_time, duration, user_ID)

foreign key: user_ID references Managers

Functional Dependencies

event_id → event_name event_date event_time duration user_ID

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

CREATE TABLE Events(

event_id **INT** not null auto_increment,

event_name **VARCHAR(20)**,

event_date **DATE**,

event_time **CHAR(5)**,

duration **INT**,

user_ID **INT** not null,

PRIMARY KEY (event_id)

FOREIGN KEY (user_ID) REFERENCES Managers

) ENGINE = InnoDB;

3.1.14 GuestEvents

Relational Model

GuestEvents(event_id,numOfJoinedGuests)

foreign key: event_id references Events

Functional Dependencies

event_id → numOfJoinedGuests

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

```
CREATE TABLE GuestEvents(
event_id                INT not null,
numOfJoinedGuests       INT not null,
PRIMARY KEY (event_id),
FOREIGN KEY (event_id) REFERENCES Events
) ENGINE = InnoDB;
```

3.1.15 GuestActivity

Relational Model

GuestActivity(event_id,type)

foreign key: event_id references GuestEvents

Functional Dependencies

event_id → type

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

CREATE TABLE GuestActivity(

event_id **INT** not null,

type **VARCHAR(20)** not null,

PRIMARY KEY (event_id),

FOREIGN KEY (event_id) REFERENCES GuestEvents

) ENGINE = InnoDB;

3.1.16 GroupTour

Relational Model

GroupTour(event_id, tour_place, tour_guide)

foreign key: event_id references GuestEvents

Functional Dependencies

event_id → tour_place tour_guide

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

CREATE TABLE GroupTour(

event_id **INT** not null,

tour_place **VARCHAR(20)** not null,

tour_guide **VARCHAR(200)** not null,

PRIMARY KEY (event_id),

FOREIGN KEY (event_id) REFERENCES GuestEvents

) ENGINE = InnoDB;

3.1.17 Movie_Show

Relational Model

Movie_Show(event_id, director, subject)

foreign key: event_id references GuestActivity

Functional Dependencies

event_id → director subject

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

CREATE TABLE Movie_Show(

event_id **INT** not null,

director **VARCHAR(20)** not null,

subject **VARCHAR(200)** not null,

PRIMARY KEY (event_id),

FOREIGN KEY (event_id) REFERENCES GuestActivity

) ENGINE = InnoDB;

3.1.18 Games

Relational Model

Games(<u>event_id</u>, game_status, prize)

foreign key: event_id references GuestActivity

Functional Dependencies

event_id → game_status prize

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

```
CREATE TABLE Games(
event_id        INT not null,
game_status     VARCHAR(20) not null,
prize           VARCHAR(50) not null,
PRIMARY KEY (event_id),
FOREIGN KEY (event_id) REFERENCES GuestActivity
) ENGINE = InnoDB;
```

3.1.19 Theater_Show

Relational Model

Theater_Show(<u>event_id</u>, director, subject)

foreign key: event_id references GuestActivity

Functional Dependencies

event_id → director subject

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

```
CREATE TABLE Theater_Show(
event_id        INT not null,
director        VARCHAR(20) not null,
subject         VARCHAR(200) not null,
PRIMARY KEY (event_id),
FOREIGN KEY (event_id) REFERENCES GuestActivity
) ENGINE = InnoDB;
```

3.1.20 Ticket

Relational Model

Ticket(<u>ticket_no</u>,<u>event_id</u>, price, user_ID)

foreign key: user_ID references Guest

foreign key: event_id references GuestEvents

Functional Dependencies

ticket_no event_id → price user_ID

Candidate Keys

{(ticket_no, event_id)}

Primary Key

(ticket_no, event_id)

Table Definition

```
CREATE TABLE Ticket(
ticket_no        INT not null auto_increment,
event_id         INT not null,
price            INT not null,
user_ID          INT not null,
PRIMARY KEY (ticket_no, event_id),
FOREIGN KEY (user_ID) REFERENCES Guests,
FOREIGN KEY (event_id) REFERENCES GuestEvents
) ENGINE = InnoDB;
```

3.1.21 Managers

Relational Model

Managers(<u>user_ID</u>, department, seniority)

foreign key: user_ID references Employees

Functional Dependencies

user_ID → department seniority

Candidate Keys

{(user_ID)}

Primary Key

(user_ID)

Table Definition

```
CREATE TABLE Managers(
user_ID          INT not null,
department       VARCHAR(20) not null,
seniority        VARCHAR(20) not null,
```

PRIMARY KEY (user_ID),

FOREIGN KEY (user_ID) REFERENCES Employees

) ENGINE = InnoDB;

3.1.22 Employees

Relational Model

Employees(user_ID, salary)

foreign key: user_ID references Hotel_Users

Functional Dependencies

user_ID → salary

Candidate Keys

{(user_ID)}

Primary Key

(user_ID)

Table Definition

CREATE TABLE Employees(

user_ID **INT** not null,

salary **INT** not null,

PRIMARY KEY (user_ID),

FOREIGN KEY (user_ID) REFERENCES Hotel_Users

);

3.1.23.Staff

Relational Model

Staff(user_ID, working_hours)

foreign key: user_ID references Employees

Functional Dependencies

user_ID → working_hours

Candidate Keys

{(user_ID)}

Primary Key

(user_ID)

Table Definition

```
CREATE TABLE Staff(
user_ID            INT not null,
working_hours      INT not null,
PRIMARY KEY (user_ID),
FOREIGN KEY (user_ID) REFERENCES Employees
) ENGINE = InnoDB;
```

3.1.24 Security_Staff

Relational Model

Security_Staff(<u>user_ID</u>, annual_leave_days)

foreign key: user_ID references Staff

Functional Dependencies

user_ID → annual_leave_days

Candidate Keys

{(user_ID)}

Primary Key

(user_ID)

Table Definition

```
CREATE TABLE Security_Staff(
user_ID                INT not null,
annual_leave_days INT,
PRIMARY KEY (user_ID),
FOREIGN KEY (user_ID) REFERENCES Staff
) ENGINE = InnoDB;
```

3.1.25 Housekeepers

Relational Model

Housekeepers(<u>user_ID</u>)

foreign key: user_ID references Staff

Functional Dependencies

None

Candidate Keys

{(user_ID)}

Primary Key

(user_ID)

Table Definition

```
CREATE TABLE Housekeepers(
user_ID        INT not null,
PRIMARY KEY (user_ID),
FOREIGN KEY (user_ID) REFERENCES Staff
) ENGINE = InnoDB;
```

3.1.26 TrainingProgram

Relational Model

TrainingProgram(event_id, trainer_name, subject)

foreign key: event_id references Events

Functional Dependencies

event_id → trainer_name subject

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

```
CREATE TABLE TrainingProgram(
event_id        INT not null,
trainer_name VARCHAR(20),
subject         VARCHAR(20),
PRIMARY KEY (event_id),
FOREIGN KEY (event_id) REFERENCES Events
) ENGINE = InnoDB;
```

3.1.27 LeaveRequestForm

Relational Model

LeaveRequestForm(form_id, request_ date, request_text, leave_date, back_date, m_user_ID, s_user_ID)

foreign key: m_user_ID references Managers(user_ID)

foreign key: s_user_ID references Security_Staff(user_ID)

Functional Dependencies

form_id → request_date request_text leave_date back_date m_user_ID s_user_ID

Candidate Keys

{(form_id)}

Primary Key

(form_id)

Table Definition

CREATE TABLE LeaveRequestForm(

form_id **INT** not null auto_increment,

request_date **DATE** not null,

request_text **VARCHAR(300)** not null,

leave_date **DATE** not null,

back_date **DATE** not null,

m_user_ID **INT** not null,

s_user_ID **INT** not null,

PRIMARY KEY (form_id),

FOREIGN KEY (m_user_ID) REFERENCES Managers(user_ID),

FOREIGN KEY (s_user_ID) REFERENCES Security_Staff(user_ID)

) ENGINE = InnoDB;

3.1.28 Clothing

Relational Model

Clothing(shop_id, numOfChangingRoom)

foreign key: shop_id references Shops

Functional Dependencies

shop_id → numOfChangingRoom

Candidate Keys

{(shop_id)}

Primary Key

(shop_id)

Table Definition

```
CREATE TABLE Clothing(
shop_id                 INT not null,
numOfChangingRoom INT,
PRIMARY KEY (shop_id),
FOREIGN KEY (shop_id) REFERENCES Shops
) ENGINE = InnoDB;
```

3.1.29 DrugStore

Relational Model

DrugStore(<u>shop_id</u>, numOfDepartments)

foreign key: shop_id references Shops

Functional Dependencies

shop_id → numOfDepartments

Candidate Keys

{(shop_id)}

Primary Key

(shop_id)

Table Definition

```
CREATE TABLE DrugStore(
shop_id                 INT not null,
numOfDepartments INT,
PRIMARY KEY (shop_id),
FOREIGN KEY (shop_id) REFERENCES Shops
) ENGINE = InnoDB;
```

3.1.30 Eatables

Relational Model

Eatables(shop_id, numOfDepartments)

foreign key: shop_id references Shops

Functional Dependencies

shop_id → numOfDepartments

Candidate Keys

{(shop_id)}

Primary Key

(shop_id)

Table Definition

```
CREATE TABLE Eatables(
shop_id                 INT not null,
numOfDepartments INT,
PRIMARY KEY (shop_id),
FOREIGN KEY (shop_id) REFERENCES Shops
) ENGINE = InnoDB;
```

3.1.31 Jewelery

Relational Model

Jewelery(shop_id)

foreign key: shop_id references Shops

Functional Dependencies

None

Candidate Keys

{(shop_id)}

Primary Key

(shop_id)

Table Definition

```
CREATE TABLE Jewelery(
shop_id        INT not null,
PRIMARY KEY (shop_id),
FOREIGN KEY (shop_id) REFERENCES Shops
) ENGINE = InnoDB;
```

3.1.32 Hotel_Users_Phone

Relational Model

Hotel_Users_Phone(user_ID, phone)

foreign key: user_ID references Hotel_Users

Functional Dependencies

None

Candidate Keys

{(user_ID)}

Primary Key

(user_ID)

Table Definition

CREATE TABLE Hotel_Users_Phone(

user_ID **INT** not null,

phone **CHAR(10),**

PRIMARY KEY (user_ID),

FOREIGN KEY (user_ID) REFERENCES Hotel_Users

) ENGINE = InnoDB;

3.1.33 Movie_Show_Stars

Relational Model

Movie_Show_Stars(event_id, stars)

foreign key: event_id references GuestActivity

Functional Dependencies

None

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

CREATE TABLE Movie_Show_Stars(

event_id **INT** not null,

stars **VARCHAR(20),**

PRIMARY KEY (event_id),

FOREIGN KEY (event_id) REFERENCES GuestActivity

) ENGINE = InnoDB;

3.1.34 Theater_Show_Players

Relational Model

Theater_Show_Players(event_id, players)

foreign key: event_id references GuestActivity

Functional Dependencies

None

Candidate Keys

{(event_id)}

Primary Key

(event_id)

Table Definition

CREATE TABLE Theater_Show_Players(

event_id **INT** not null,

players **VARCHAR(20),**

PRIMARY KEY (event_id),

FOREIGN KEY (event_id) REFERENCES GuestActivity

) ENGINE = InnoDB;

3.2 Relationships

3.2.1 assign_hkeeper

Relational Model

assign_hkeeper(order_id, h_user_ID, m_user_ID, status)

foreign key: order_id references Order

foreign key: h_user_ID references Housekeepers

foreign key: m_user_ID references Managers

Functional Dependencies

order_id h_user_ID → m_user_ID status

Candidate Keys

{(order_id, h_user_ID)}

Table Definition

```
CREATE TABLE assign_hkeeper(
order_id      INT,
h_user_ID     INT,
m_user_ID     INT,
status        VARCHAR(20)
PRIMARY KEY (order_id, h_user_ID),
FOREIGN KEY (order_id) REFERENCES Order,
FOREIGN KEY (h_user_ID) REFERENCES Housekeepers,
FOREIGN KEY (m_user_ID) REFERENCES Managers
) ENGINE = InnoDB;
```

3.2.2 buy_good

Relational Model

buy_good(user_ID, good_id)

foreign key: user_ID references Guests

foreign key: good_id references Goods

Functional Dependencies

None

Candidate Keys

{(user_ID, good_id)}

Table Definition

CREATE TABLE buy_good(

user_ID **INT**,

good_id **INT**,

PRIMARY KEY (user_ID, good_id),

FOREIGN KEY (user_ID) REFERENCES Guests,

FOREIGN KEY (good_id) REFERENCES Goods

) ENGINE = InnoDB;

3.2.3 has_good

Relational Model

has_good(good_id, shop_id)

foreign key: good_id references Goods

foreign key: shop_id references Shops

Functional Dependencies

good_id → shop_id

Candidate Keys

{(good_id)}

Table Definition

CREATE TABLE has_good(

```
good_id        INT,
shop_id        INT,
PRIMARY KEY (good_id),
FOREIGN KEY (good_id) REFERENCES Goods,
FOREIGN KEY (shop_id) REFERENCES Shops
) ENGINE = InnoDB;
```

3.2.4 shops_in_a

Relational Model

shops_in_a(shop_id, building_no, building_name)

foreign key: shop_id references Shops

foreign key: (building_no, building_name) references Building

Functional Dependencies

shop_id → building_no, building_name

Candidate Keys

{(shop_id)}

Table Definition

```
CREATE TABLE shops_in_a(
shop_id        INT,
building_no    INT,
building_name VARCHAR(20),
PRIMARY KEY (shop_id),
FOREIGN KEY (shop_id) REFERENCES Shops,
FOREIGN KEY (building_no, building_name) REFERENCES Building
) ENGINE = InnoDB;
```

3.2.5 for_room

Relational Model

for_room(reservation_id, room_no, room_location)

foreign key: reservation_id references Reservation

foreign key: (room_no, room_location) references Room

Functional Dependencies

None

Candidate Keys

{(reservation_id, room_no, room_location)}

Table Definition

```
CREATE TABLE for_room(
reservation_id INT,
room_no        INT,
room_location VARCHAR(20),
PRIMARY KEY (reservation_id),
FOREIGN KEY (reservation_id) REFERENCES Reservation,
FOREIGN KEY (room_no, room_location) REFERENCES Room
) ENGINE = InnoDB;
```

3.2.6 assign_security

Relational Model

assign_security(building_no, building_name, s_user_ID, m_user_ID)

foreign key: (building_no, building_name) references Building

foreign key: s_user_ID references Security_Staff

foreign key: m_user_ID references Managers

Functional Dependencies

building_no building_name s_user_ID → m_user_ID

Candidate Keys

{(building_no, building_name, s_user_ID)}

Table Definition

CREATE TABLE assign_security(

building_no **CHAR(3)**,

building_name **VARCHAR(20)**,

s_user_ID **INT**,

m_user_ID **INT**,

PRIMARY KEY (building_no, building_name, s_user_ID),

FOREIGN KEY (building_no, building_name) REFERENCES Building,

FOREIGN KEY (s_user_ID) REFERENCES Security_Staff,

FOREIGN KEY (m_user_ID) REFERENCES Managers

) ENGINE = InnoDB;

3.2.7 create_events

Relational Model

create_events(<u>event_id,</u> city, region, street, user_ID)

foreign key: (city, region, street) references Location

foreign key: event_id references Events

foreign key: user_ID references Managers

Functional Dependencies

event_id → city region street user_ID

Candidate Keys

{(event_id)}

Table Definition

```
CREATE TABLE create_events(
event_id      INT,
city          VARCHAR(20),
region        VARCHAR(20),
street        VARCHAR(20),
user_ID       INT,
PRIMARY KEY (event_id),
FOREIGN KEY (city, region, street) REFERENCES Location,
FOREIGN KEY (event_id) REFERENCES Events,
FOREIGN KEY (user_ID) REFERENCES Managers
) ENGINE = InnoDB;
```

3.2.8 evaluate_leave_request

Relational Model

evaluate_leave_request(<u>form_id,</u> user_ID, status)

foreign key: form_id references LeaveRequestForm

foreign key: user_ID references Managers

Functional Dependencies

form_id → user_ID status

Candidate Keys

{(form_id)}

Table Definition

```
CREATE TABLE evaluate_leave_request(
form_id        INT,
user_ID        INT,
status         VARCHAR(20),
PRIMARY KEY (form_id),
FOREIGN KEY (form_id) REFERENCES LeaveRequestForm,
FOREIGN KEY (user_ID) REFERENCES Managers
) ENGINE = InnoDB;
```

3.2.9 evaluates

Relational Model

evaluates(<u>event_id, s_user_ID</u>, m_user_ID, status)

foreign key: event_id references TrainingProgram

foreign key: s_user_ID references Staff

foreign key: m_user_ID references Managers

Functional Dependencies

event_id s_user_ID → m_user_ID status

Candidate Keys

{(event_id, s_user_ID)}

Table Definition

```
CREATE TABLE evaluates(
event_id       INT,
s_user_ID      INT,
m_user_ID      INT
status          VARCHAR(20),
```

PRIMARY KEY (event_id, s_user_ID),

FOREIGN KEY (event_id) REFERENCES TrainingProgram,

FOREIGN KEY (s_user_ID) REFERENCES Staff,

FOREIGN KEY (m_user_ID) REFERENCES Managers

) ENGINE = InnoDB;

3.2.10 apply_to

Relational Model

apply_to(event_id, user_ID)

foreign key: event_id references TrainingProgram

foreign key: user_ID references Staff

Functional Dependencies

None

Candidate Keys

{(event_id, user_ID)}

Table Definition

CREATE TABLE apply_to(

event_id **INT**,

user_ID **INT**,

PRIMARY KEY (event_id, user_ID),

FOREIGN KEY (event_id) REFERENCES TrainingProgram,

FOREIGN KEY (user_ID) REFERENCES Staff

) ENGINE = InnoDB;

3.2.11 fill

Relational Model

fill(form_id, user_ID)

foreign key: form_id references LeaveRequestForm

foreign key: user_ID references Security_Staff

Functional Dependencies

form_id → user_ID

Candidate Keys

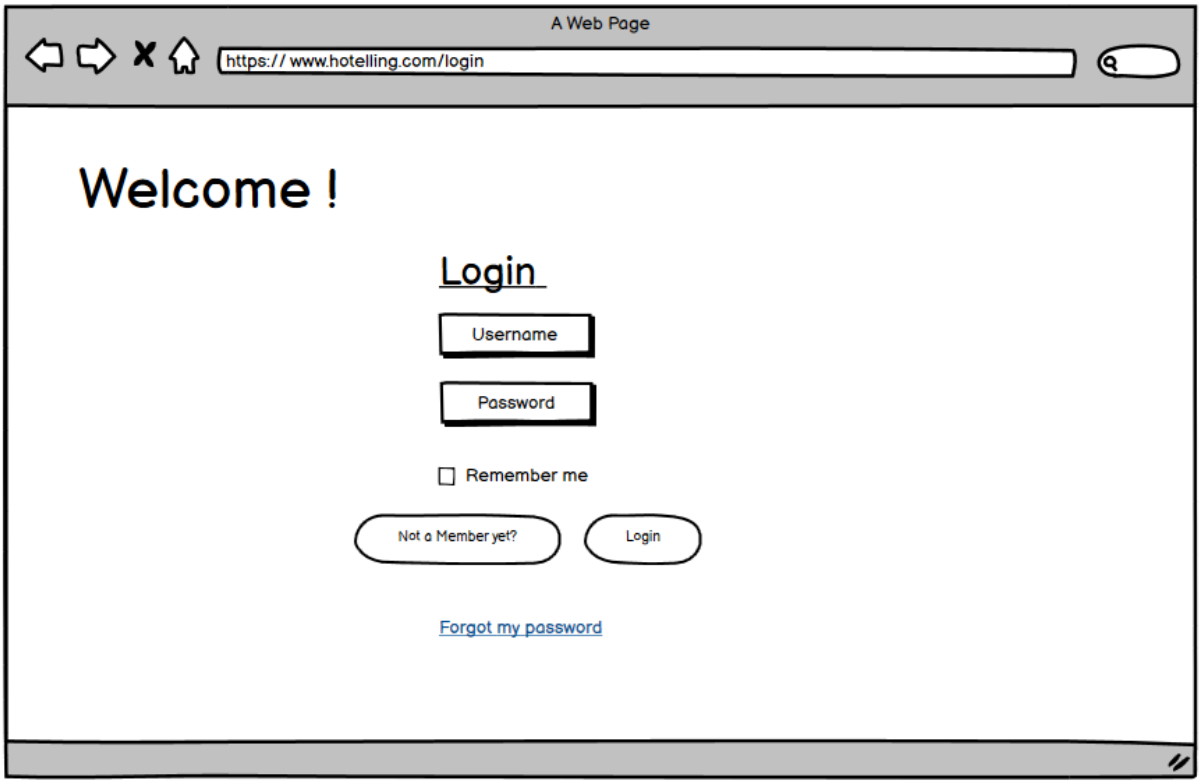
{(form_id)}

Table Definition

```
CREATE TABLE fill(
form_id        INT,
user_ID        INT,
PRIMARY KEY (form_id),
FOREIGN KEY (form_id) REFERENCES LeaveRequestForm,
FOREIGN KEY (user_ID) REFERENCES Security_Staff
) ENGINE = InnoDB;
```

4. User Interface Design and SQL Statements

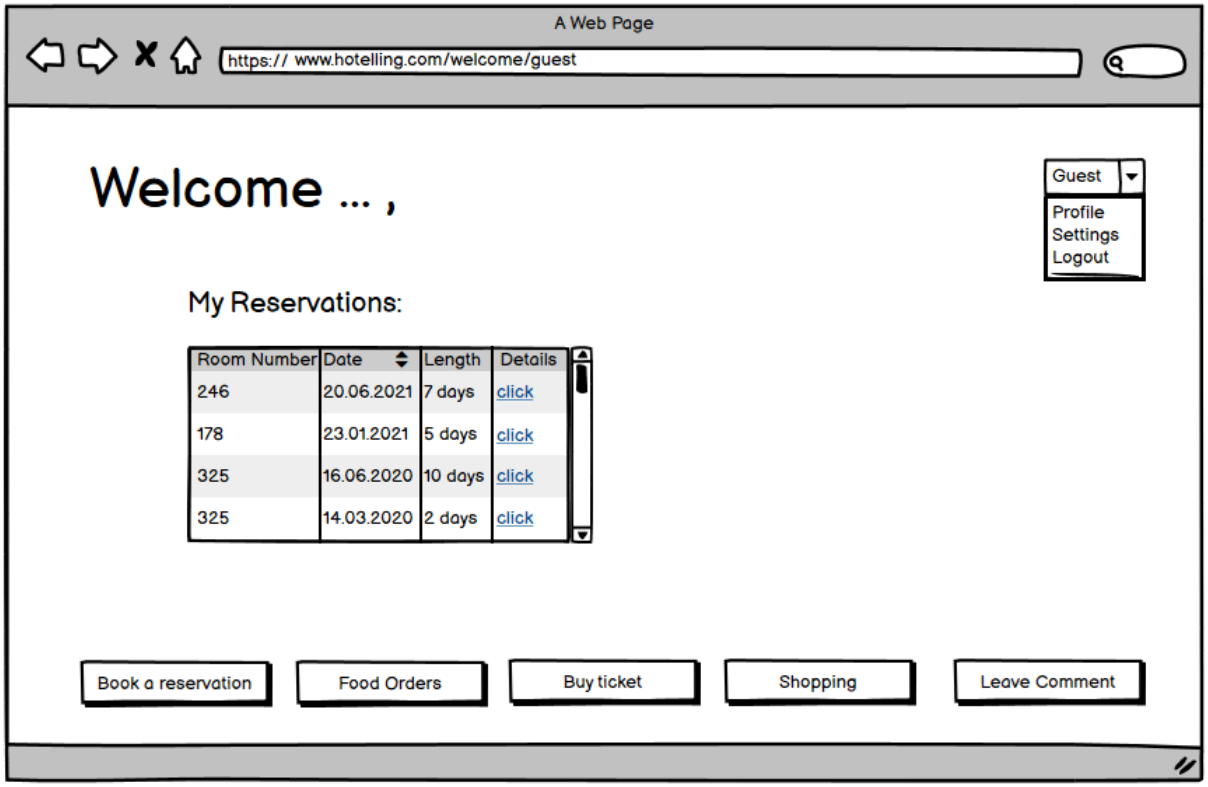
4.1 Login Page



Login as user:

```
select * from Hotel_Users
where user_name = @username and password = @password
```

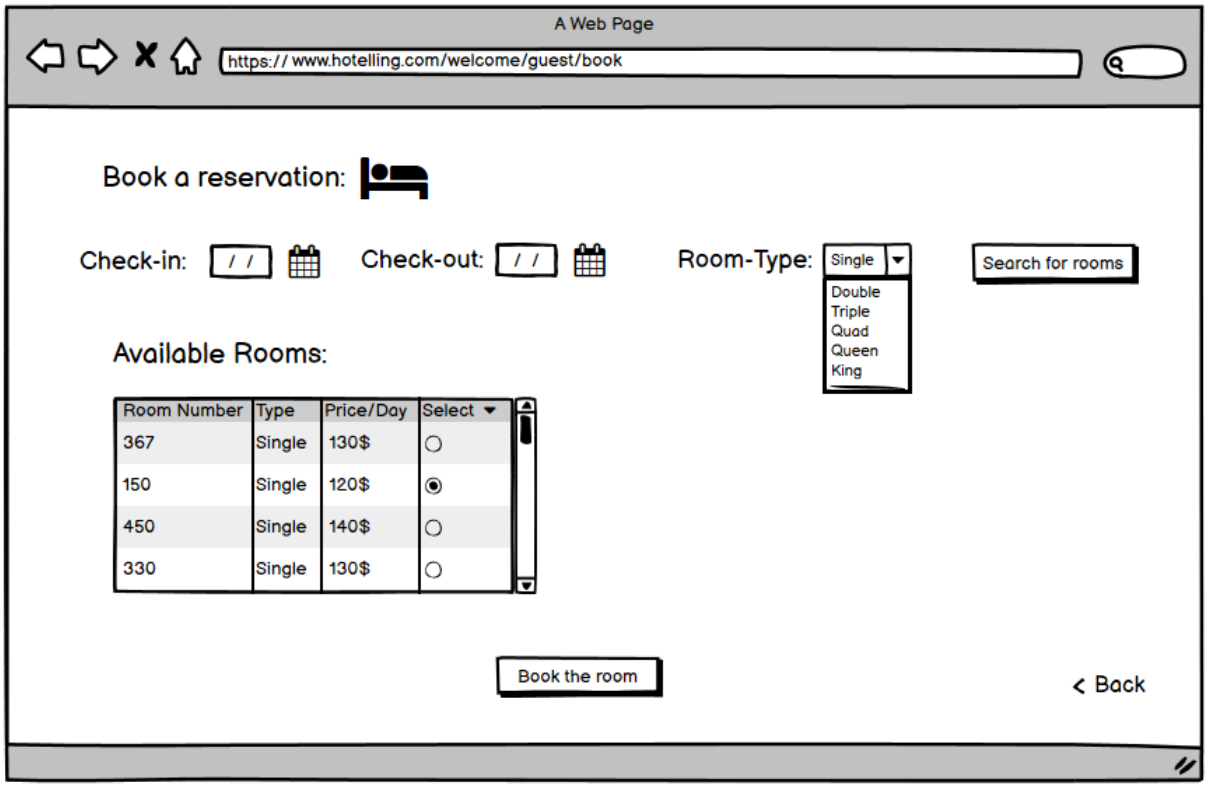
4.2 Guest Page



My Reservations:

```
select room_no, check_in_date,
        (check_out_date - check_in_date) as length
from for_room natural join book_reservation
where user_ID = @user_ID
```

4.3 Book a Reservation Page



Available Rooms:

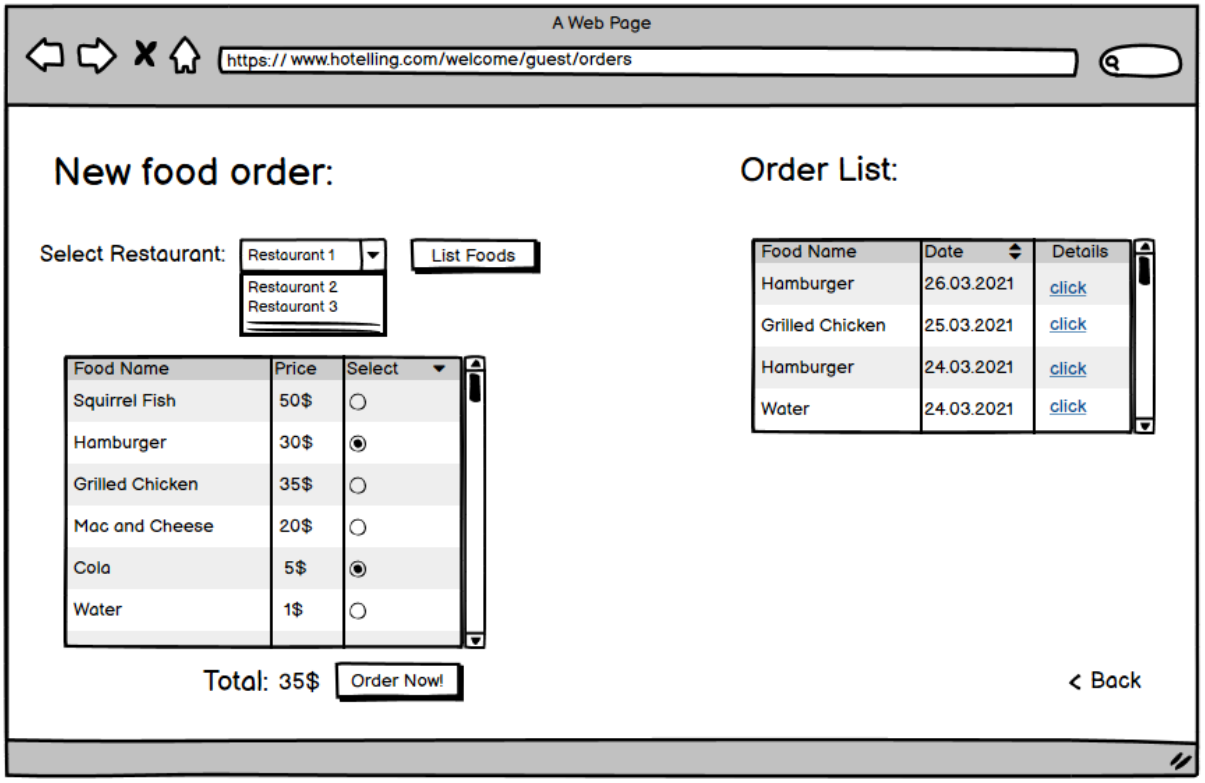
select room_no, room_type, room_price
from Room
where room_availability = ‘available’

Book a Reservation:

insert into Reservation
values (reservation_id: 0, getdate(), @check_in_date, @check_out_date, @numOfStayers, numOfRooms: 1, @price)

update Room
set room_availability = ‘not available’
where
    (select * from Room natural join for_room
    where reservation_id = @reservation_id)

4.4 Food Orders Page



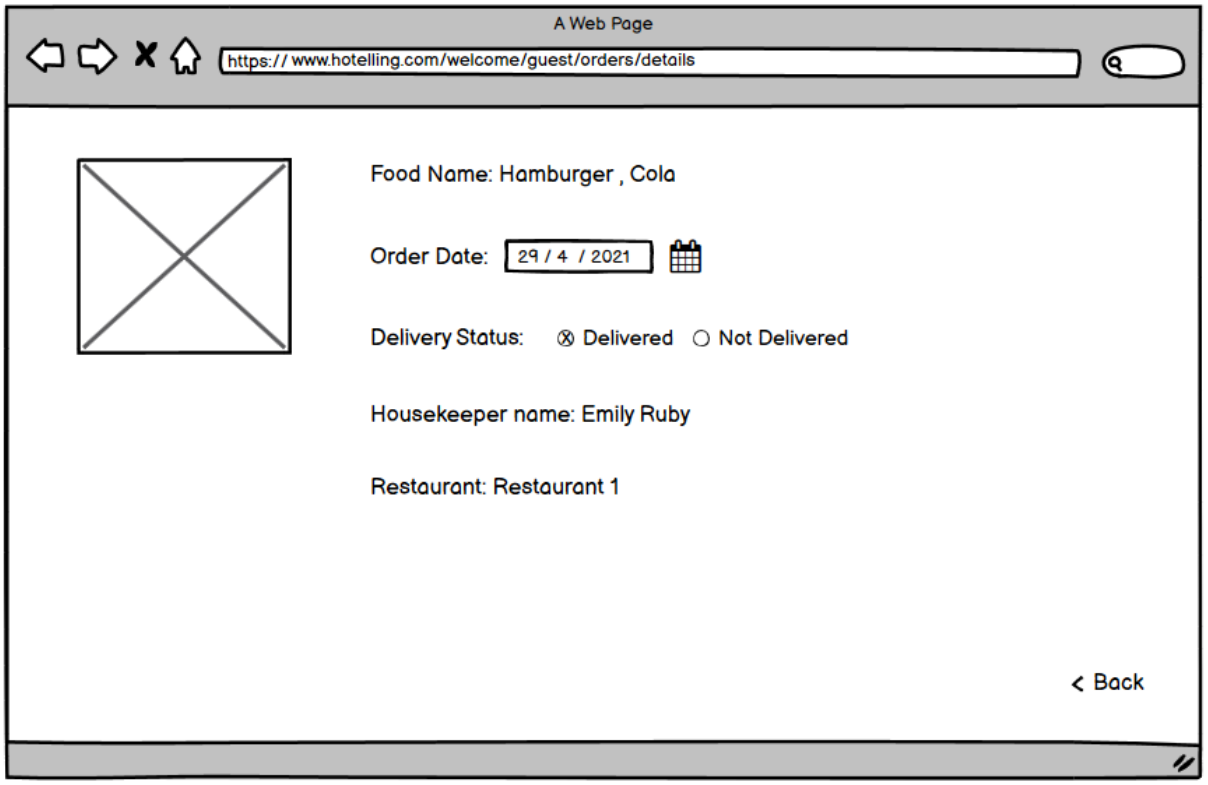
New Food Order:
select food_name, food_price
from Food natural join Order natural join Restaurant
where restaurant_id = @restaurant_id

Order Now:
insert into Order
values(order_id: 0, getdate(), @price, @food_id, @user_ID)

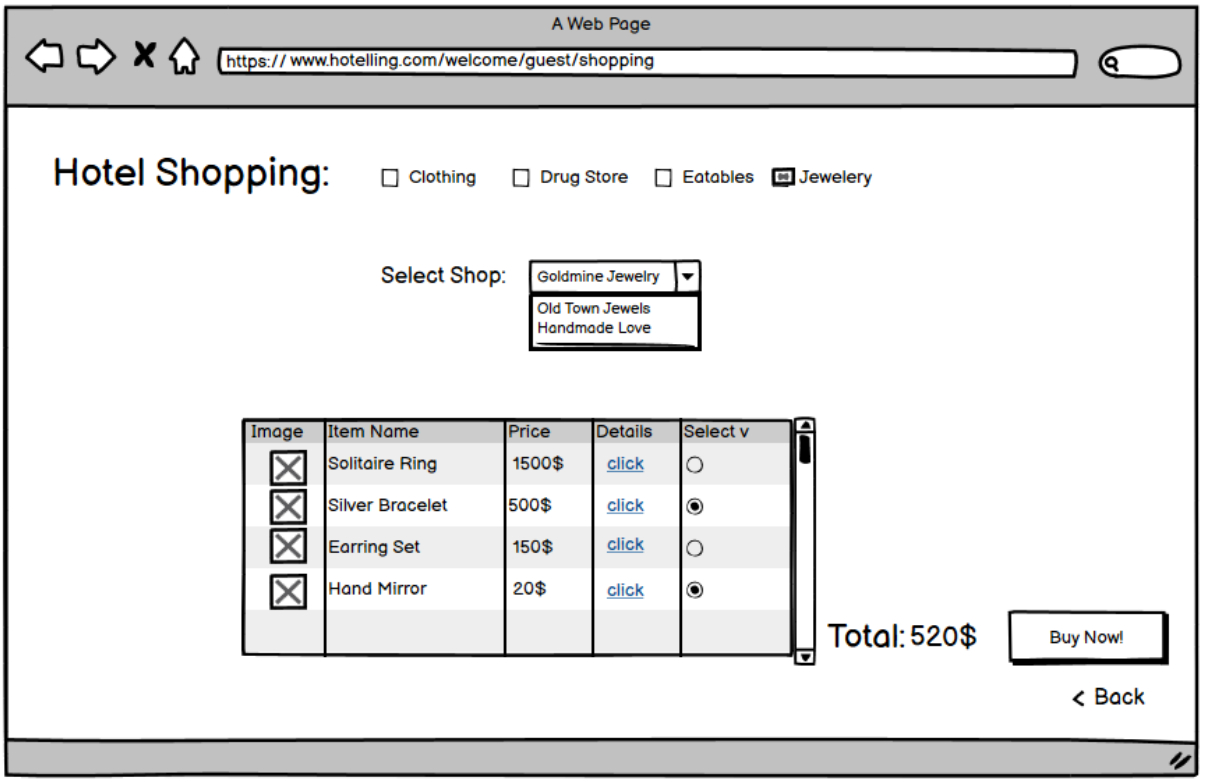
Order List:
select food_name, order_date
from Food natural join Order
where user_ID = @user_ID

Order Detail:
select * from Order
where order_id = @order_id

4.5 Order Details Page



4.6 Hotel Shop Page



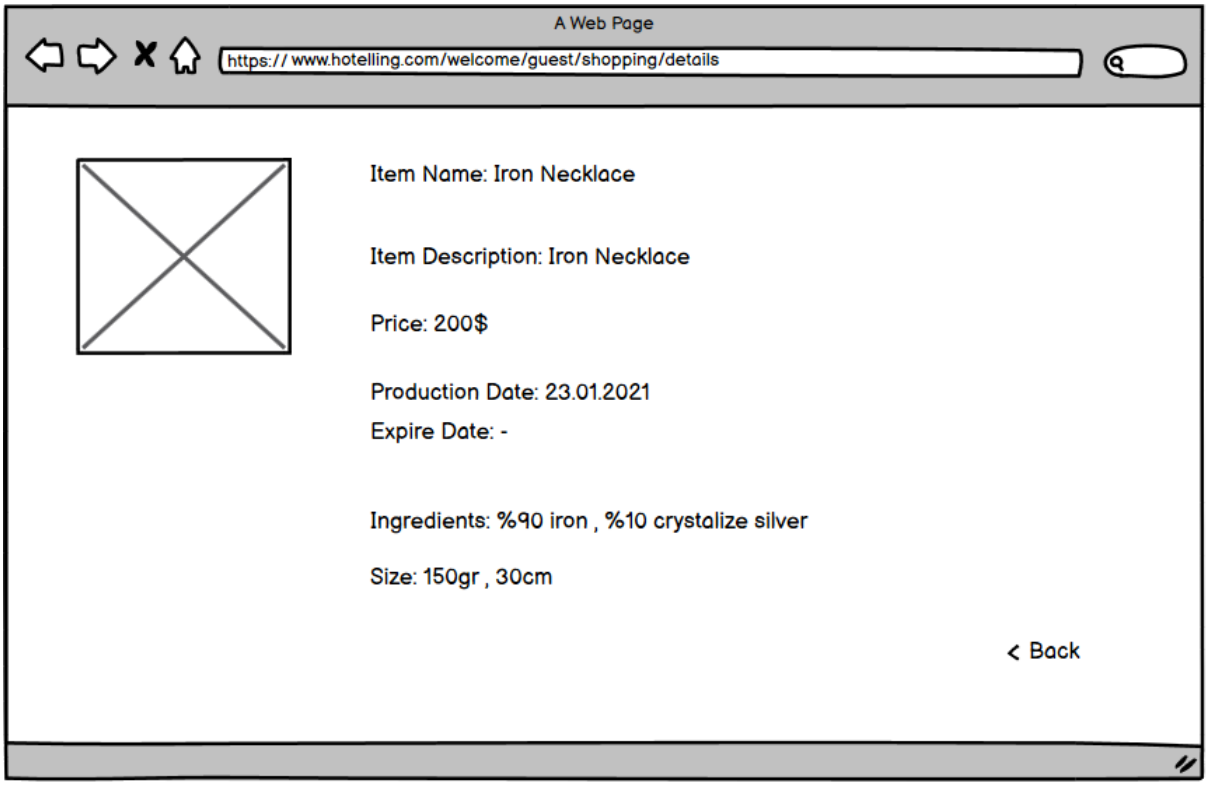
Select Shop:
select *
from Jewelery
where shop_name = @shop_name

List Goods:
select good_name, price
from Goods natural join Shops natural join has_good
where shop_id = @shop_id and shop_name = @shop_name

Buy Now:
insert into buy_good
values(@user_id, @good_id)

Shop Detail:
select * from Goods
where good_id = @good_id

4.7 Shopping Details Page



5. Implementation Plan

For our Hotel Database Management System web server, InnoDB will be chosen as a database engine and MySQL will be used for database management. For front-end development of the web-site, Javascript and CSS languages and for back-end development of the web-site, PHP language will be used to implement.

6. Webpage

The status of the project can be checked at the attached GitHub repository link.

https://github.com/velicanmert/Inn-Manager

The project website will be updated when required.

https://handesey.github.io/Inn-Manager/