CS 353 Database Systems

Project Design Report

Group 7

Travel Agency

Name	Last Name	ID
Irmak	Akyeli	21803690
Tuva Tanay	Işıksal	21503011
Melike	Demirci	21702346
Kimya	Ghasemlou	21801412

Table of Contents

1. Introduction

In this report we are going to share revisited E/R model, relational schemas, user interfaces, SQL queries for some functionalities and implementation decisions of our travel agency project.

2. Revised E/R Model

We corrected our E/R diagram with the feedback of our TA as following:

- Missing total participation of activities to places_activities relation fixed.
- Corrected the relation feedback as it is not a weak entity relation.
- Added a description attribute to the tour entity.
- Fixed one to many and many to many relations with the arrow heads.
- Added a reservation entity to make our database more efficient.
- Added a booking entity as the users can also make room bookings independently.

We also made the following changes to make easier implementation of queries as well as them being more efficient:

- Changed c_age attribute to c_bdate in order to not update it each year.
- Added wallet attribute to the customer to enable the payment.
- Added g_rating attribute to guide for holding the avg evaluation points the guide has.
- Added t_rating attribute to tour for holding the avg evaluation points the tour obtained.
- Changed the name of the table user to general_user in order to not complicate the implementation.
- Added username to general_user as the users should have their name and an username stored.

- Added is_accepted and explanation to the relation book_flights as the employee should be able accept and decline the booking.
- Added is_accepted and explanation to the relation book_room as the employee should be able accept and decline the booking.
- Added is_accepted and explanation to the relation reserves as the employee should be able accept and decline the reservation and a tuple at the reservation table is created if it is accepted.
- Made the tour entity connect directly to the places_activities relation instead of goes_to relation that seemed unnecessary.
- Removed the d_available attribute of guide entity as it depends on the tour and it cannot be stored like that.
- Added a text attribute to the feedback relation to actually hold the feedback.

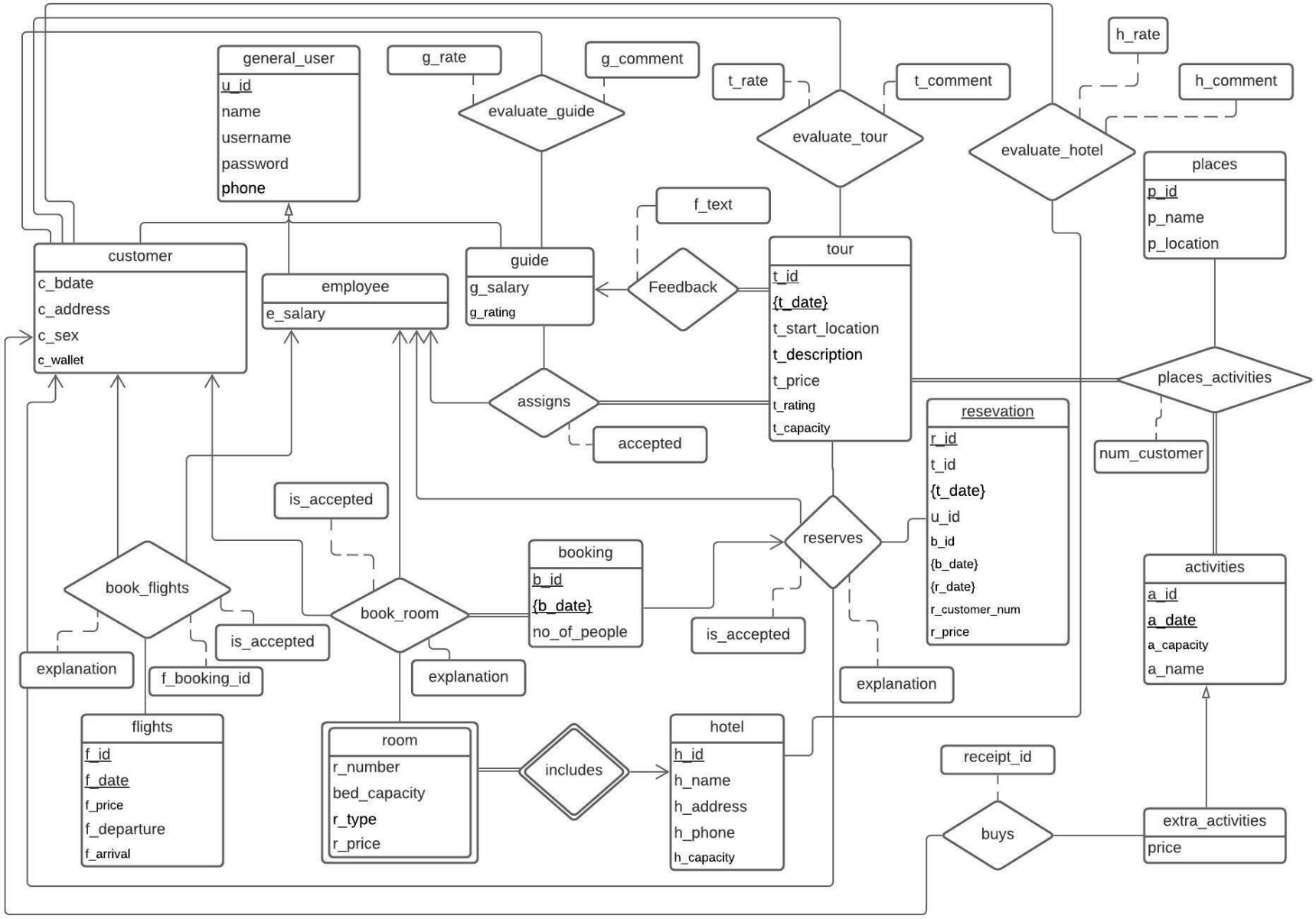
general_user
u_id
name
username
password
phone
g_rate
g_comment
evaluate_guide
t_rate
t_comment
evaluate_tour
h_rate
h_comment
evaluate_hotel
places
p_id
p_name
p_location
f_text
customer
c_bdate
c_address
c_sex
c_wallet
employee
e_salary
guide
g_salary
g_rating
Feedback
tour
t_id
{t_date}
t_start_location
t_description
t_price
t_rating
t_capacity
places_activities
assigns
accepted
resevation
r_id
t_id
{t_date}
u_id
b_id
{b_date}
{r_date}
r_customer_num
r_price
num_customer
is_accepted
reserves
booking
b_id
{b_date}
no_of_people
book_flights
book_room
is_accepted
is_accepted
activities
a_id
a_date
a_capacity
a_name
explanation
f_booking_id
explanation
explanation
flights
f_id
f_date
f_price
f_departure
f_arrival
room
r_number
bed_capacity
r_type
r_price
includes
hotel
h_id
h_name
h_address
h_phone
h_capacity
receipt_id
buys
extra_activities
price

Figure 1

3. Relation Schemas

3.1. User

- **Relational Model:**

  generalUser(u_id, u_name, pw, phone)

- **Functional Dependencies:**

  u_id → name, username, pw, phone

  username → u_id

  phone→ u_id

- **Candidate Keys:** {(u_id),(username),(phone)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE generalUser (u_id INT NOT NULL,
        name CHAR(50),
        username CHAR(20),
        pw VARCHAR(50),
        phone NUMERIC(11,0),
        PRIMARY KEY (u_id)) ENGINE=innodb;
```

3.2. Customer

- **Relational Model:**

  customer(u_id, name, username, c_bdate, c_address, c_sex, c_wallet, pw, phone)

- **Functional Dependencies:**

  u_id → name, username, c_bdate, c_address, c_sex, c_wallet, pw, phone

- **Candidate Keys:** {(u_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE customer (u_id INT,
        name CHAR(50),
        username CHAR(20),
        c_bdate Date,
        c_address VARCHAR(50),
        c_sex CHAR(10),
        c_wallet INT,
        pw NUMERIC(11,0),
        phone INT,
        PRIMARY KEY (u_id)) ENGINE=innodb;
```

3.3. Employee

- **Relational Model:**

  employee(u_id, name, username, phone, pw, e_salary)

- **Functional Dependencies:**

  u_id → name, username, phone, pw

- **Candidate Keys:** {(u_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE employee (u_id INT,
        name CHAR(50),
        username CHAR(20),
        phone INT,
        pw NUMERIC(11,0),
        e_salary INT,
        PRIMARY KEY (u_id)) ENGINE=innodb;
```

3.4. Guide

- **Relational Model:**

  guide(u_id, name, username, phone, pw, g_salary, g_points, g_rating)

- **Functional Dependencies:**

  u_id → name, username, phone, pw

- **Candidate Keys:** {(u_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE guide(u_id INT,
        name VARCHAR(50),
        username CHAR(20),
        phone INT,
        pw NUMERIC(11,0),
        g_salary INT,
        g_points INT,
        g_rating INT,
        PRIMARY KEY (u_id)) ENGINE=innodb;
```

3.5. Booking

- **Relational Model:**

  booking(b_id, b_start_date, b_end_date, no_of_people)

- **Functional Dependencies:**

  b_id → b_start_date, b_end_date, no_of_people

- **Candidate Keys:** {(b_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE booking (b_id INT,
        b_start_date DATE,
        b_end_date DATE,
        no_of_people INT,
        PRIMARY KEY (b_id, b_start_date, b_end_date)) ENGINE=innodb;
```

3.6. Tour

- **Relational Model:**

  tour(t_id, t_start_date, t_end_date, t_start_location, t_price, t_capacity)

- **Functional Dependencies:**

  t_id, t_start_date, t_end_date → t_start_location, t_price, t_capacity

- **Candidate Keys:** {(t_id, t_start_date, t_end_date)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE tour (t_id INT,
        t_start_date DATE,
        t_end_date DATE,
        t_start_location VARCHAR(10),
        t_description VARCHAR(50),
        t_price INT,
        t_rating INT,
        t_capacity INT,
        PRIMARY KEY (t_id,t_start_date, t_end_date))
        ENGINE=innodb;
```

3.7. Places

- **Relational Model:**

  places(p_id, p_name, p_location)

- **Functional Dependencies:**

  p_id → p_name, p_location

- **Candidate Keys:**{(p_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE places (p_id INT,
        p_name VARCHAR(8),
        p_location VARCHAR(50),
        PRIMARY KEY (p_id)) ENGINE=innodb;
```

3.8. Activities

- **Relational Model:**

  activities(<u>a_id</u>, <u>a_date</u>, a_capacity, a_name)

- **Functional Dependencies:**

  a_id, a_date → a_date, a_capacity, a_name

- **Candidate Keys:** {(a_id, a_date)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE activities (a_id INT,
        a_date Date,
        a_capacity INT,
        a_name VARCHAR(20),
        PRIMARY KEY (a_id, a_date)) ENGINE=innodb;
```

3.9. Extra Activities

- **Relational Model:**

  extra_activities(<u>a_id</u>, <u>a_date</u>, a_capacity, a_name, price)

- **Functional Dependencies:**

  a_id, a_date → a_capacity, a_name, price

- **Candidate Keys:**{(a_id, a_date)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE extra_activities (a_id INT,
        a_date Date,
        a_capacity INT,
        a_name VARCHAR(20),
        price INT,
        PRIMARY KEY (a_id, a_date)) ENGINE=innodb;
```

3.10. Hotel

- **Relational Model:**

  hotel(h_id, h_name, h_address, h_phone, h_capacity)

- **Functional Dependencies:**

  h_id → h_name, h_address, h_phone, h_capacity

- **Candidate Keys:** {h_id}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE hotel(h_id INT,
        h_name VARCHAR(50),
        h_address VARCHAR(50),
        h_phone NUMERIC(11,0),
        h_capacity INT,
        PRIMARY KEY (h_id)) ENGINE=innodb;
```

3.11. Room

- **Relational Model:**

  room(h_id, r_number, bed_capacity, r_type, r_price)

  h_id : FK to hotel(h_id)

- **Functional Dependencies:**

  h_id, r_number → bed_capacity, r_type, r_price

- **Candidate Keys:**{(h_id, r_number)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE room(h_id INT,
        r_number INT,
        bed_capacity INT,
        r_type CHAR(12),
        r_price INT,
        PRIMARY KEY (h_id, r_number),
        FOREIGN KEY (h_id) REFERENCES hotel(h_id) ) ENGINE=innodb;
```

3.12. Flights

- **Relational Model:**

  flights(f_id, f_date, f_price, f_departure, f_arrival)

- **Functional Dependencies:**

  f_id, f_date → f_price, f_departure, f_arrival

- **Candidate Keys:**{(f_id, f_date)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE flights(f_id INT,
        f_date DATE,
        f_price INT,
        f_departure VARCHAR(50),
        f_arrival VARCHAR(50),
        PRIMARY KEY (f_id, f_date)) ENGINE=innodb;
```

3.13. Reservation

- **Relational Model:**

reservation(r_id, t_id, t_start_date, t_end_date, u_id, b_id, b_start_date, b_end_date,

r_start_date, r_end_date)

- **Functional Dependencies:**

  r_id→ t_id, t_start_date, t_end_date, u_id, b_id, b_start_date, b_end_date,

  r_start_date, r_end_date

- **Candidate Keys:** {(r_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE reservation(r_id INT,
        t_id INT,
        t_start_date DATE,
        t_end_date DATE,
        u_id INT,
        b_id INT,
        b_start_date DATE,
        b_end_date DATE,
        r_start_date DATE,
        r_end_date DATE,
        r_price INT,
        r_customer_num INT,
        PRIMARY KEY (r_id),
        FOREIGN KEY (t_id, t_start_date, t_end_date) REFERENCES
        tour(t_id, t_start_date, t_end_date),
        FOREIGN KEY (u_id) REFERENCES customer(u_id),
        FOREIGN KEY (b_id, b_start_date, b_end_date) REFERENCES
        booking(b_id, b_start_date, b_end_date) ) ENGINE=innodb;
```

3.14. Evaluate Guide

- **Relational Model:**

  evaluate_guide(g_id, c_id, g_comment, g_rate)

- **Functional Dependencies:**

  g_id, c_id → g_comment, g_rate

- **Candidate Keys:** {(g_id), (c_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE  evaluate_guide ( g_id INT,
        c_id INT,
        g_comment CHAR(50),
        g_rate INT,
        PRIMARY KEY(g_id, c_id),
        FOREIGN KEY(g_id) REFERENCES guide(u_id),
        FOREIGN KEY(c_id) REFERENCES customer(u_id) )
        ENGINE=innodb;
```

3.15. Evaluate Tour

- **Relational Model:**

  evaluate_tour(t_id, t_start_date, t_end_date, c_id, t_comment, t_rate)

- **Functional Dependencies:**

  t_id, t_start_date, t_end_date → c_id, t_comment, t_rate

- **Candidate Keys:** {(t_id, t_start_date, t_end_date)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE  evaluate_tour ( t_id INT,
        t_start_date DATE,
        t_end_date DATE,
        c_id INT,
        t_comment CHAR(50),
        t_rate INT,
        PRIMARY KEY(t_id, t_start_date, t_end_date,  c_id),
        FOREIGN KEY(t_id, t_start_date, t_end_date) REFERENCES
        tour(t_id, t_start_date, t_end_date),
        FOREIGN KEY(c_id) REFERENCES customer(u_id) )
        ENGINE=innodb;
```

3.16. Evaluate Hotel

- **Relational Model:**

  evaluate_hotel(h_id, c_id, h_comment, h_rate)

- **Functional Dependencies:**

  h_id, c_id→ h_comment, h_rate

- **Candidate Keys:**{(h_id, c_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE  evaluate_hotel ( h_id INT,
            c_id INT,
            h_comment CHAR(50),
            h_rate INT,
            PRIMARY KEY(h_id, c_id),
            FOREIGN KEY(h_id) REFERENCES hotel(h_id),
            FOREIGN KEY(c_id) REFERENCES customer(u_id) )
            ENGINE=innodb;
```

3.17. Places Activities

- **Relational Model:**

  places_activities(t_id, t_start_date, t_end_date, a_id, a_date, p_id,

  num_customer)

- **Functional Dependencies:**

  t_id, t_start_daate, t_end_date, a_id, a_date, p_id → num_customer

- **Candidate Keys:**{(t_id, t_start_daate, t_end_date, a_id, a_date, p_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE  places_activities ( t_id INT,
            t_start_date Date,
            t_end_date Date,
            a_id INT,
            a_date DATE,
            p_id INT,
            num_customer INT,
            PRIMARY KEY(t_id, t_start_date, t_end_date, a_id, a_date, p_id),
            FOREIGN KEY(t_id, t_start_date, t_end_date) REFERENCES
            tour(t_id, t_start_date, t_end_date),
            FOREIGN KEY(a_id, a_date) REFERENCES activities(a_id, a_date),
            FOREIGN KEY(p_id) REFERENCES places(p_id) )
            ENGINE=innodb;
```

3.18. Buys

- **Relational Model:**

  buys(receipt_id, a_id, a_date, price, c_id)

- **Functional Dependencies:**

  receipt_id → a_id, a_date, price, c_id

- **Candidate Keys:** {(receipt_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE  buys ( receipt_id  INT,
                a_id INT,
                a_date DATE,
                price INT,
                c_id INT,
                PRIMARY KEY(receipt_id),
                FOREIGN KEY(a_id, a_date) REFERENCES extra_activities(a_id,
                a_date),
                FOREIGN KEY(c_id) REFERENCES customer(u_id) )
                ENGINE=innodb;
```

3.19. Reserves

- **Relational Model:**

  reserves(r_id, t_id, t_start_date, t_end_date, b_start_date, b_end_date, b_id, e_id,

  c_id, explanation, is_accepted)

- **Functional Dependencies:**

  r_id → r_id, t_id, t_start_date, t_end_date, b_start_date, b_end_date, b_id, e_id,

  c_id, explanation, is_accepted

- **Candidate Keys:**{(r_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE  reserves ( r_id INT,
                t_id INT,
                t_start_date DATE,
                t_end_date DATE,
                b_start_date DATE,
                b_end_date DATE,
                b_id INT,
                e_id INT,
                c_id INT,
                explanation CHAR(50),
                is_accepted BOOLEAN,
                PRIMARY KEY(r_id, e_id, c_id),
                FOREIGN KEY(r_id) REFERENCES reservation(r_id),
                 FOREIGN KEY(t_id, t_start_date, t_end_date) REFERENCES
               tour(t_id, t_start_date, t_end_date),
               FOREIGN KEY(b_id, b_start_date, b_end_date) REFERENCES
               booking(b_id, b_start_date, b_end_date),
                FOREIGN KEY(e_id) REFERENCES employee(u_id),
               FOREIGN KEY(c_id) REFERENCES customer(u_id) )
               ENGINE=innodb;
```

3.20. Feedback

- **Relational Model:**

  feedback(<u>t_id, t_start_date, t_end_date, g_id</u>, f_text)

- **Functional Dependencies:**

  t_id, t_start_date, t_end_date, g_id → f_text

- **Candidate Keys:**

  {(t_id, t_start_date, t_end_date, g_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE  feedback ( t_id INT,
                t_start_date DATE,
                t_end_date DATE,
                g_id INT,
                f_text VARCHAR(250),
                PRIMARY KEY(t_id, t_start_date, t_end_date, g_id),
                FOREIGN KEY(t_id, t_start_date, t_end_date) REFERENCES
                tour(t_id, t_start_date, t_end_date),
                FOREIGN KEY(g_id) REFERENCES guide(u_id))ENGINE=innodb;
```

3.21. Includes

- **Relational Model:**

  includes(h_id, r_number)

- **Functional Dependencies:**

  h_id, r_number→ h_id, r_number (trivial)

- **Candidate Keys:** {(h_id, r_number)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE  includes ( h_id INT,
                r_number INT,
                PRIMARY KEY(h_id, r_number),
                FOREIGN KEY(h_id) REFERENCES hotel(h_id),
                 FOREIGN KEY(h_id, r_number) REFERENCES room(h_id,
                r_number) ) ENGINE=innodb;
```

3.22. Assign

- **Relational Model:**

  assign(t_id, t_start_date, t_end_date, g_id, e_id, accepted)

- **Functional Dependencies:**

  t_id, t_start_date, t_end_date, g_id, e_id → accepted

- **Candidate Keys:** {(t_id, t_start_date, t_end_date, g_id, e_id)}

- **Normal Form:** BCNF
- **Table Definition:**

```
CREATE TABLE  assign ( t_id INT,
        t_start_date DATE,
        t_end_date DATE,
        g_id INT,
        e_id INT,
        accepted BOOLEAN,
        PRIMARY KEY(t_id, t_start_date, t_end_date, g_id, e_id),
        FOREIGN KEY(t_id,t_start_date, t_end_date) REFERENCES
        tour(t_id, t_start_date, t_end_date),
        FOREIGN KEY(e_id) REFERENCES employee(u_id) ,
        FOREIGN KEY(g_id) REFERENCES guide(u_id) )
        ENGINE=innodb;
```

3.23. Book Room

- **Relational Model:**

  book_room(b_id, b_start_date, b_end_date, c_id, h_id, e_id, r_number, is_accepted, explanation)

- **Functional Dependencies:**

  b_id, b_start_date, b_end_date, c_id, h_id, e_id, r_number → is_accepted,

  explanation

- **Candidate Keys: {(**b_id, b_start_date, b_end_date, c_id, h_id, e_id, r_number)**}**
- **Normal Form:** BCNF
- **Table Definition:**

```
CREATE TABLE  book_room ( b_id INT,
        b_start_date DATE,
        b_end_date DATE,
        c_id INT,
        h_id INT,
        e_id INT,
        r_number INT,
        is_accepted BOOLEAN,
        explanation CHAR(250),
```

```
PRIMARY KEY(b_id, b_start_date, b_end_date, h_id, r_number, c_id,
e_id),
FOREIGN KEY(b_id, b_start_date, b_end_date) REFERENCES
booking(b_id, b_start_date, b_end_date),
FOREIGN KEY(h_id) REFERENCES hotel(h_id),
FOREIGN KEY(h_id, r_number) REFERENCES room(h_id,
r_number),
FOREIGN KEY(c_id) REFERENCES customer(u_id),
FOREIGN KEY(e_id) REFERENCES employee(u_id) )
ENGINE=innodb;
```

3.24. Book Flights

- **Relational Model:**

  book_flights(f_booking_id, f_id, f_date, c_id, e_id, is_accepted, explanation)

- **Functional Dependencies:**

  f_booking_id → f_id, f_date, c_id, e_id, is_accepted, explanation

- **Candidate Keys:** {(f_booking_id)}

- **Normal Form:** BCNF

- **Table Definition:**

```
CREATE TABLE book_flights ( f_id INT,
        f_date DATE,
        c_id INT,
        e_id INT,
        f_booking_id INT,
        is_accepted BOOLEAN,
        explanation CHAR(250),
        PRIMARY KEY(f_booking_id),
        FOREIGN KEY(f_id,f_date) REFERENCES flights(f_id, f_date),
        FOREIGN KEY(c_id) REFERENCES customer(u_id),
        FOREIGN KEY(e_id) REFERENCES employee(u_id) )
        ENGINE=innodb;
```

4. User Interface and SQL Queries

4.1. Login

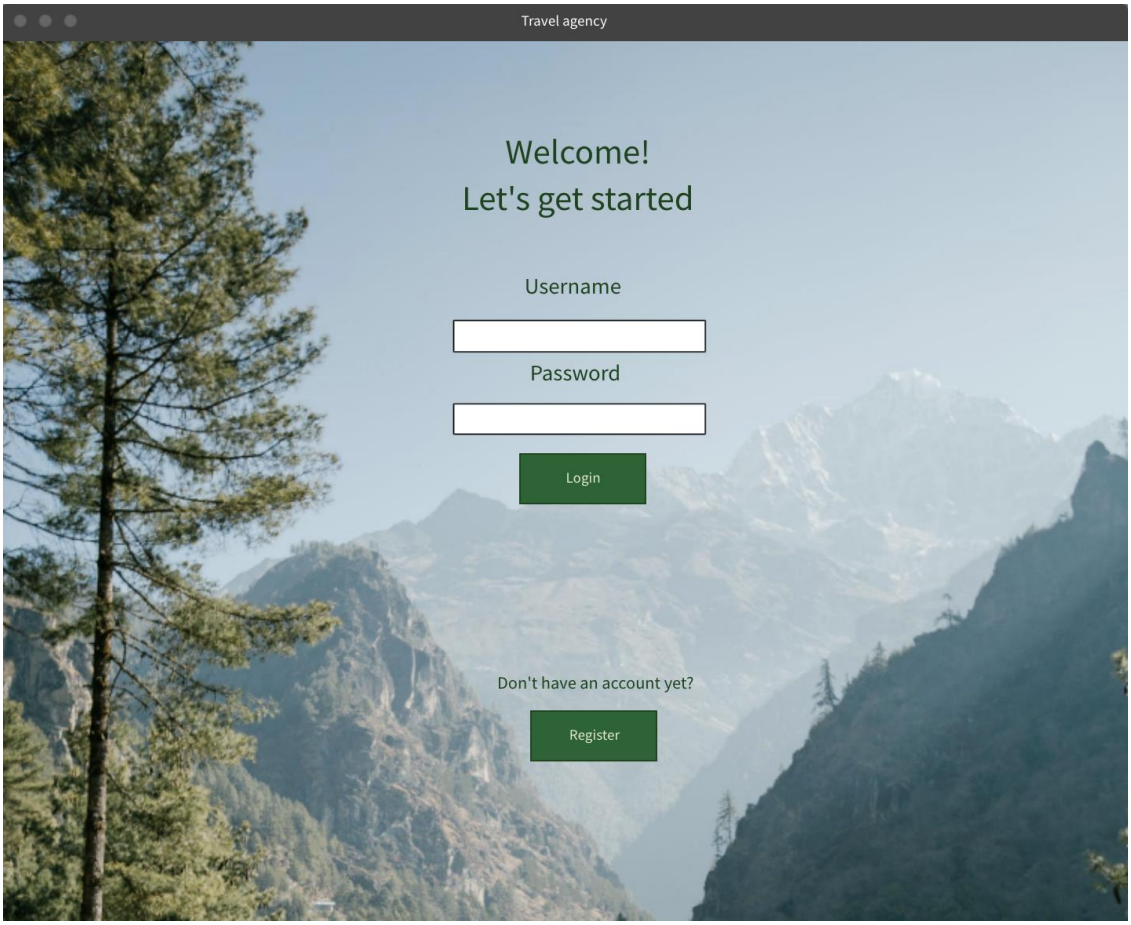

Figure 2: Login Screen [2], [3].

Process: All the users enter the portal via entering their usernames and passwords.

Inputs: @username, @password

SQL Query:

SELECT username, pw

FROM generalUser

WHERE username = @username AND pw = @password

4.2. Register

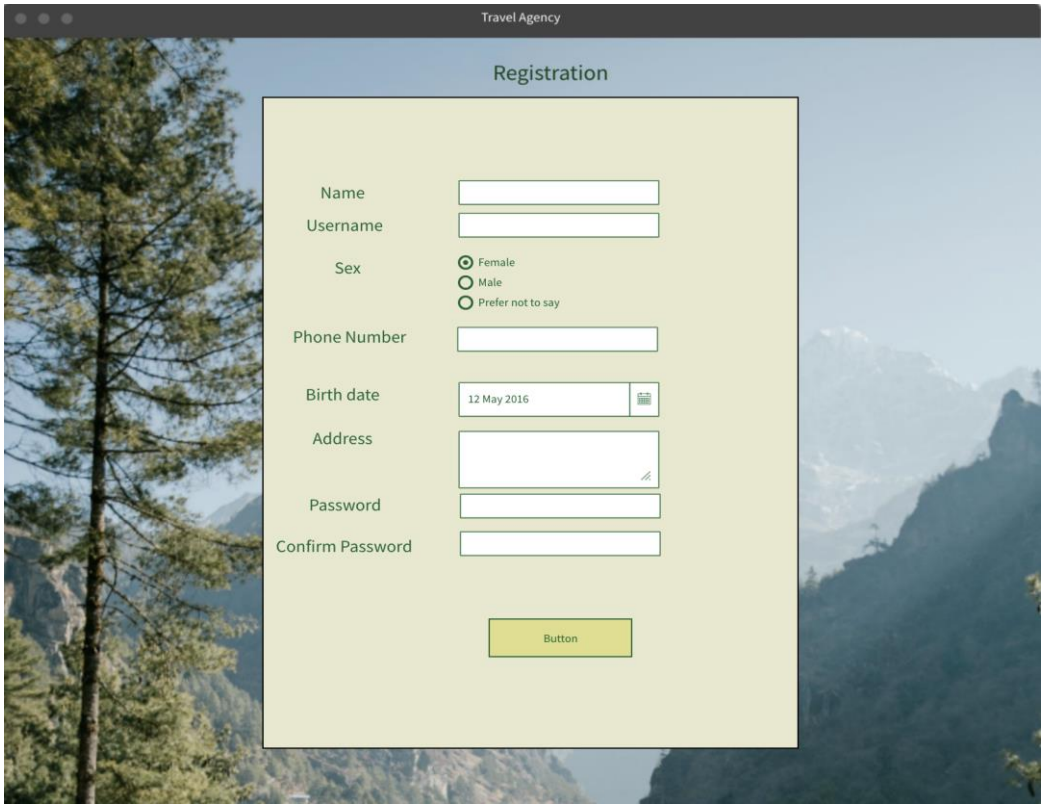

Figure 3: Registration Screen for Customer [2], [3].

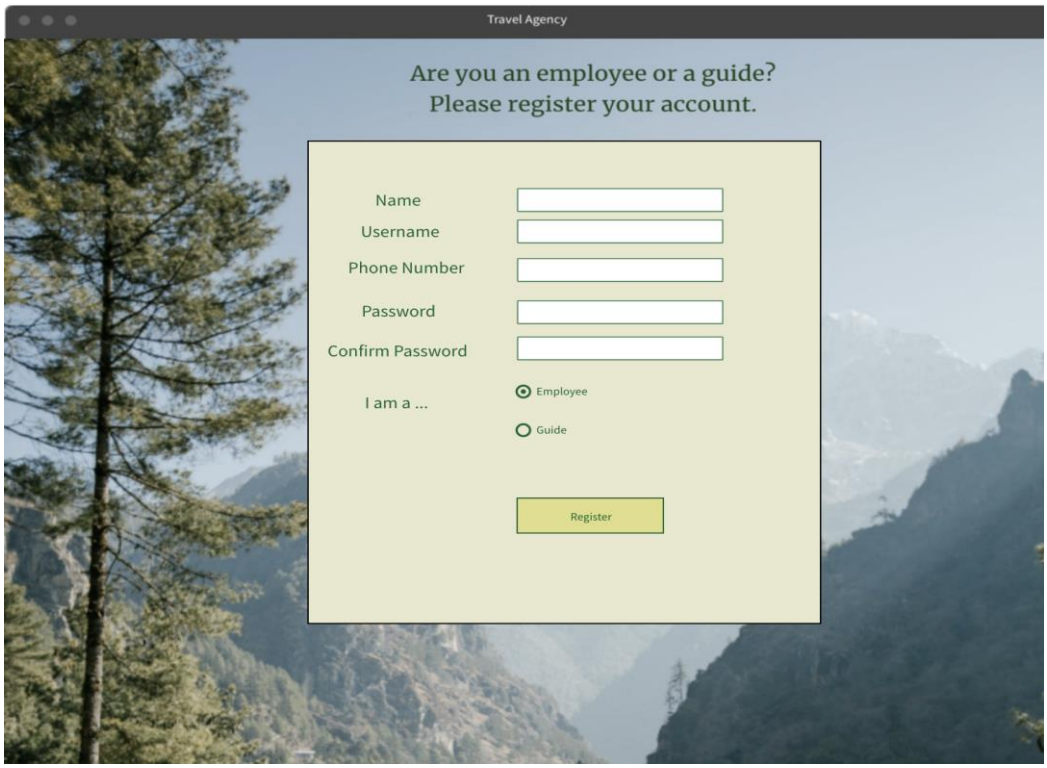

Figure 4: Registration Screen for employee or guide [2], [3].

We have three user types that are customer, employee and guide. The customer and guide/employee have different registration interfaces as they require different information.

Process: The user enters their information in required places and registers.

Register For customer:

Inputs: @name, @username, @password, @confirmpassword, @sex, @birthdate, @address, @phonenumber,

SQL Query:

INSERT INTO customer(name, username, c_bdate, c_address, c_sex, pw, phone)
VALUES(@name, @username, @birthdate, @address, @sex, @password, @phonenumber)

Register For employee:

Inputs: @name, @username, @phonenumber, @password, @confirmpassword

SQL Query:

INSERT INTO employee(name, username, phone, pw)
VALUES(@name, @username, @phone, @password)

Register For guide:

Inputs: @name, @username, @phonenumber, @password, @confirmpassword

SQL Query:

INSERT INTO guide(name, username, phone, pw)
VALUES(@name, @username, @phone, @password)

4.3. Filter

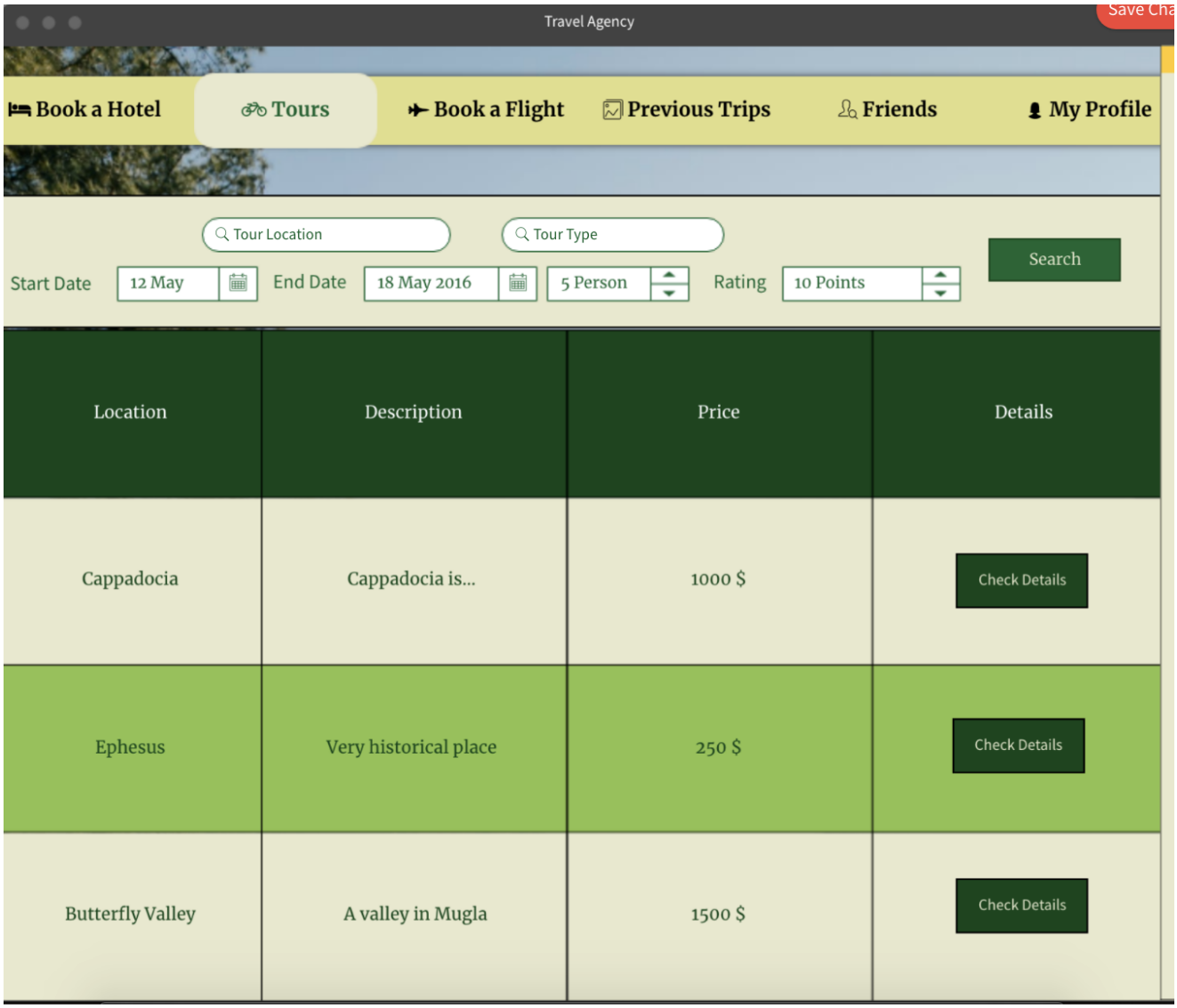

Figure 5: Tours Screen [2], [3].

Process: The Tours page shows all existing tours as default. If the user wants, they can filter the search with related dates, number of persons, rating, location or types. If the search consists of more than one filter, our implementation will give the null input values that do not change the result so that we don't have unknown values.

Inputs: @start_date @end_date @people_no @location @rating @desc

SQL Queries:

Filter with date:

SELECT *

FROM tour

WHERE t_start_date >= @start_date and t_end_date <= @end_date

Filter with people number:

SELECT *

FROM tour

WHERE t_capacity >= @people_no

Filter with location:

SELECT *

FROM tour

WHERE t_start_location == @location

Filter with rating:

SELECT *

FROM tour

WHERE t_rating >= @rating

Filter with description:

SELECT *

FROM tour

WHERE t_description like “%desc%”

Filter with all:

SELECT *

FROM tour

WHERE t_start_date >= @start_date and t_end_date <= @end_date and t_capacity >= @people_no and t_start_location == @location and t_rating >= @rating and t_description like “%desc%”

4.4 Indicate The Number of People

Process: In every related UI there is a number of people selection tools that have a default value 1 that can be changed by the user.

Inputs: @selected_res

SQL Queries:

SELECT r_customer_num

FROM reservation

WHERE r_id == @r_id

4.5. Tour Selection

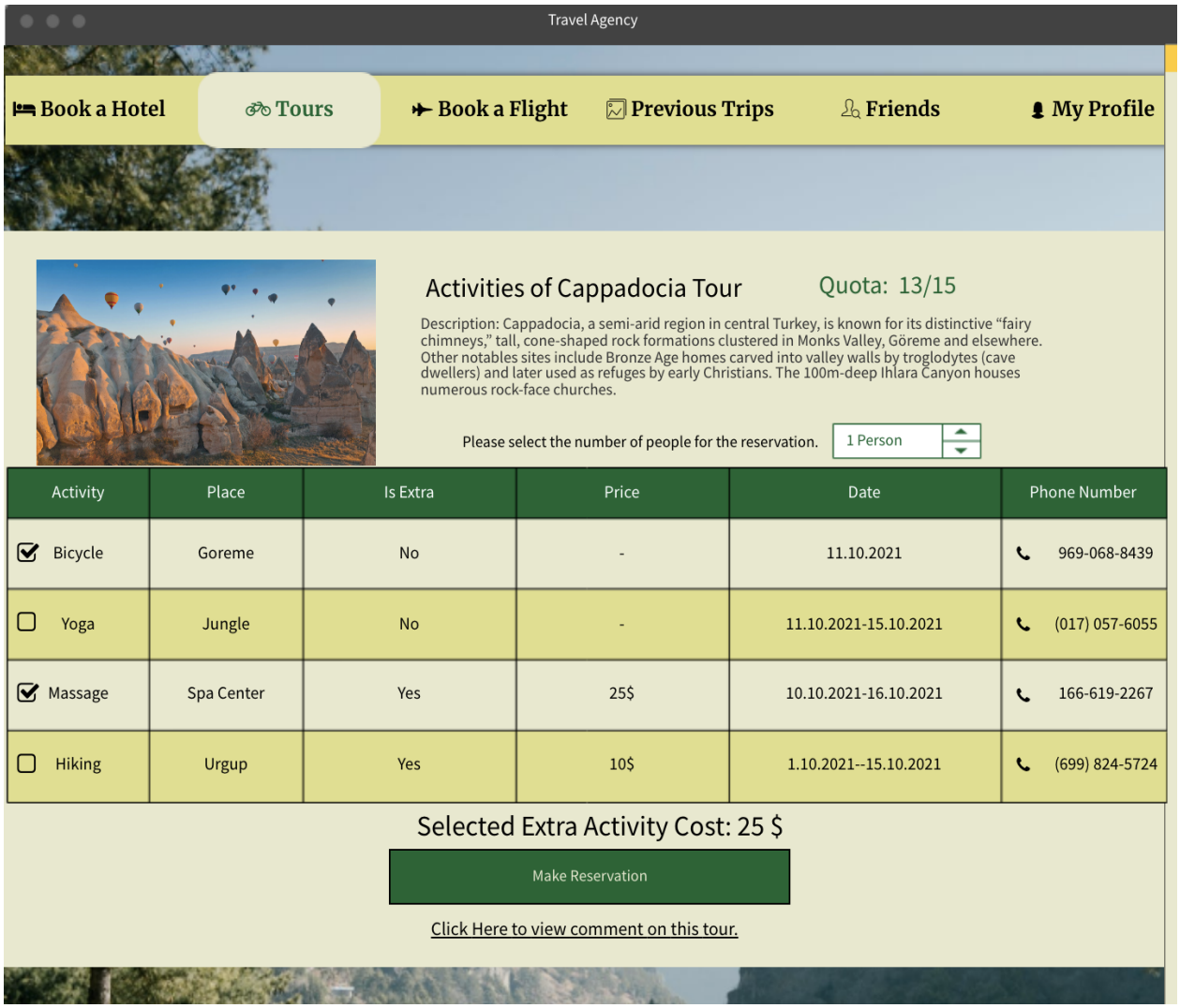

Figure 6: Tours Screen with Activity Details [2], [3].

Process: When check details of the previous UI is clicked, our implementation will provide the inputs for the detail page.

Inputs: @selected_id, @start_date, @end_date

SQL Queries:

SELECT *

FROM tour

WHERE t_id == @selected_id and t_start_date >= @start_date and t_end_date <= @end_date

4.6. List Activities

Process: As it can be seen on the previous UI in the details page there will be a table of all activities provided for the tour. Extra activities will have a price where the included ones don’t. All the activities require reservations but included ones do not require any payment.

Inputs: @selected_tour

SQL Queries:

SELECT a.*

FROM tour natural join places_activities as p, activities as a

WHERE p.a_,d = a.a_id and p.a_date = a.a_date and t_id == @selected_tour

4.7. Make Payment

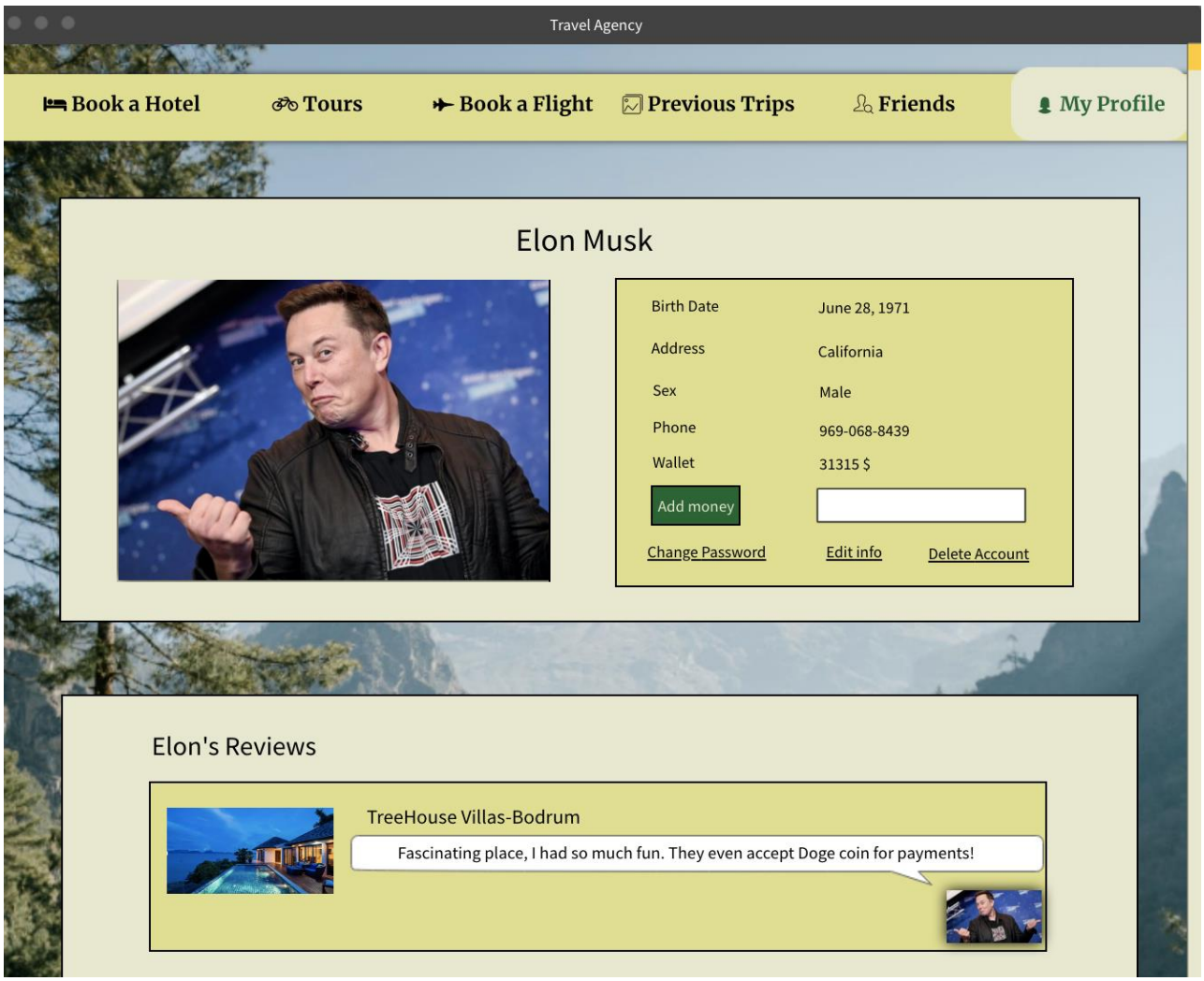

Figure 7: Customer Profile Screen [2], [3].

Process: Every customer can see their wallet and add money to their account from their profile and when making a payment, they are required to have the amount of tour, room or activity.

Inputs: @selected_extra_activity @u_id

SQL Queries:

UPDATE customer SET wallet = wallet - (SELECT price FROM extra_activity AS E WHERE E.a_id = @selected_extra_activity.a_id)

4.8. Give Feedback

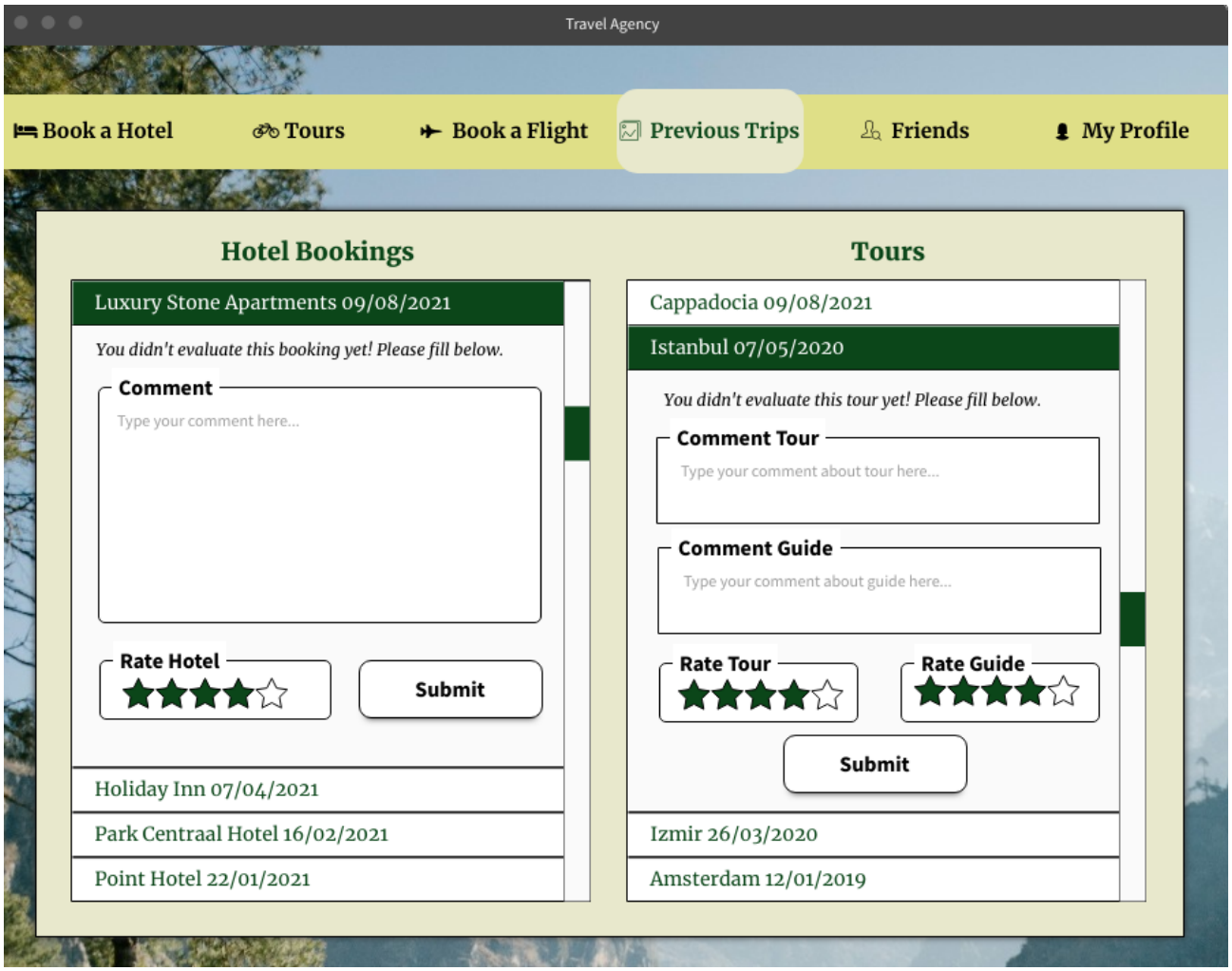

Figure 8: Feedback Screen [2], [3].

Process: After a customer takes a tour and their guide/guides, he/she can evaluate their experience of the tour in the related UI and their evaluations can be found on their profiles. These evaluation points will also affect the rating of the tours and guides.

Inputs: @c_id @seleceted_tour @g_id @tour_rate @guide_rate @tour_comment @guide_comment

SQL Queries:

INSERT INTO evaluate_tour(t_id, t_start_date, t_end_date, c_id, t_comment, t_rate)

VALUES(@seleceted_tour.t_id,@seleceted_tour.t_start_date,@seleceted_tour.t_end_date, c_id, @t_comment, @t_rate)

INSERT INTO evaluate_guide(g_id, c_id, g_comment, g_rate)

VALUES (@g_id,@c_id, @guide_comment, @guide_rate)

4.9 Additional

As we proposed to have additional feature to buy flights along with the tour we have the following UI:

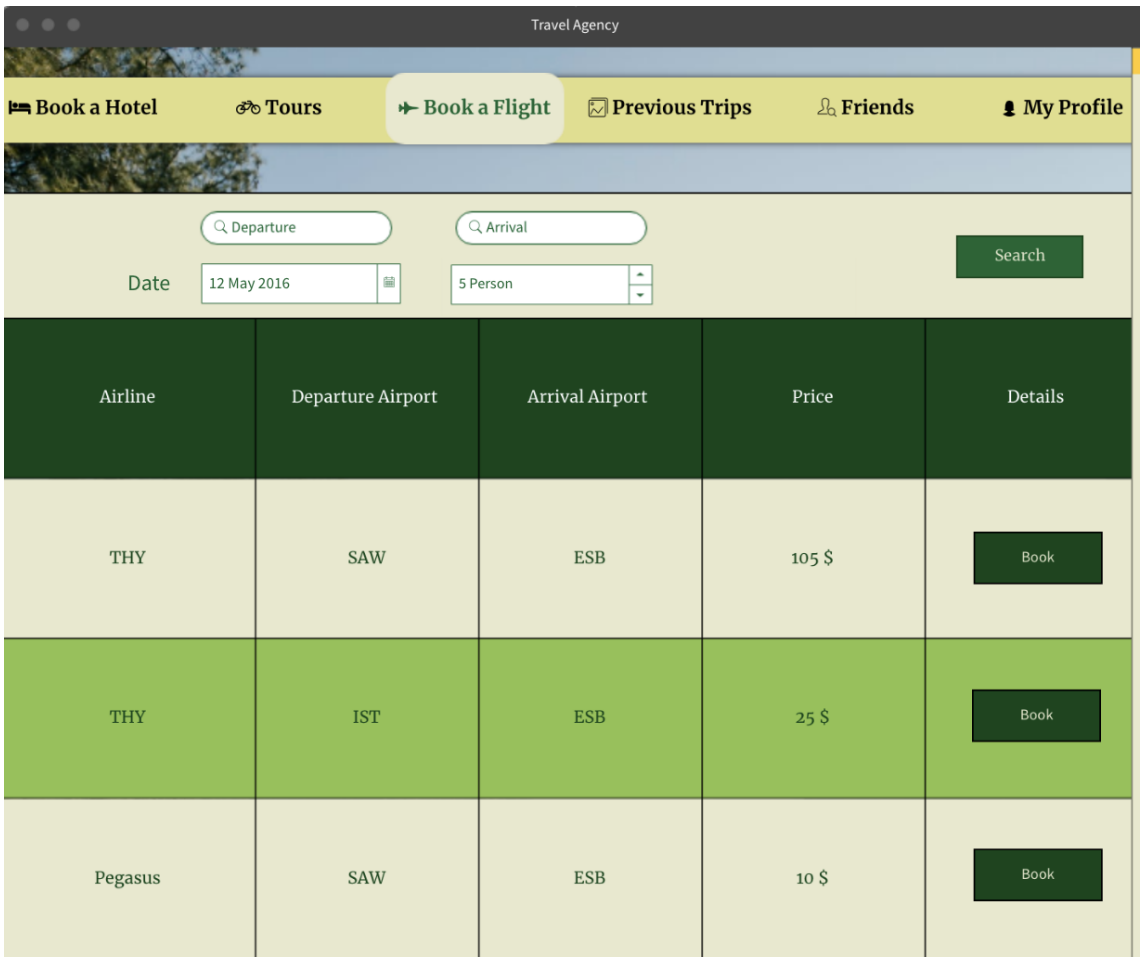

Figure 9: Book Flight Screen [2], [3].

Process: Customers can check flights with available filters like date and place. They can make the payment via our portal too.

Inputs: @date, @arrival, @departure

SQL Queries:

To filter:

SELECT *,

FROM flights,

WHERE f_date = @date AND f_arrival = @arrival AND f_departure = @departure

Reservation for flight:

Input = @date, @arrival, @departure

INSERT INTO book_flights(f_date, f_departure, f_arrival)

VALUES(@date, @departure, @arrival)

UPDATE customer SET wallet = wallet - (SELECT f_price FROM flights AS E WHERE E.f_id = @selected_flight.f_id)

5. Implementation

In this web based database application project, MySQL will be used for database management. Bootstrap, HTML, CSS, Javascript and JQuery for the front-end and Python's Django framework will be used for the back-end.

6. References

[1] “Lucidchart” www.lucidchart.com/
[2] “Wire Frame Pro” wireframepro.mockflow.com
[3] “Color Hunt” colorhunt.co