Bilkent University

Department of Computer Engineering

CS 353 Term Project

Travel Agency Data Management System

Final Report

Project Group Members:

Zübeyir Bodur - 21702382

Funda Tan - 21801861

Emine Ezgi Saygılı - 21802871

Abdul Razak Daher Khatib - 21801340

TA: Mustafa Can Çavdar

Table of Contents

1. Overview

The system will provide the users the opportunity to plan their trips. The system enables users to reserve a tour by themselves or through an agent of the company, later they can edit and update the reservations. The reservations can be made for the hotel rooms or the tours. The tour will include several sightseeing places that are usually visited by tourists. Along with that, they will also have the ability to participate in activities like festivals and concerts. After the tour is over the user will be able to review the tour or the guide and rate them. The guide will also have the right to add feedback about the tour. To make things easier and beneficial for the agency we added multiple features to make the system more convenient and easier to use. For example, a discount can be applied to any reservation made by a customer, which can help the agency promote its tours. In summary, we provide in our system what most travel agencies would need, from reservations to activities and sightseeing places, to discounts for the reservations.

2. Final ER Diagram

We have revised our E/R model based on the feedback given on the Design Report, as well as the problems encountered in the implementation, as follows:

- In order to show which activities were chosen in a reservation, a marked_activity relation was added, between a TourReservation and Activity.
- The entry_fee was removed from SightseeingPlaces.
- The age property is changed to birth_date in the Users table
- The tour_group table was not necessary, so it was removed.
- Tour_days was changed to tour_end_date.
- Now both reservations have a start date and end date.
- Primary keys are named consistently. For example, the ID attribute was renamed to be u_id, instead of ID.
- Discount_type is removed, discount_start date is moved to the relations instead. To ensure that discount_start_date will be defined only when the rows participate in relations, the necessary check constraints are added (See 5.2.)

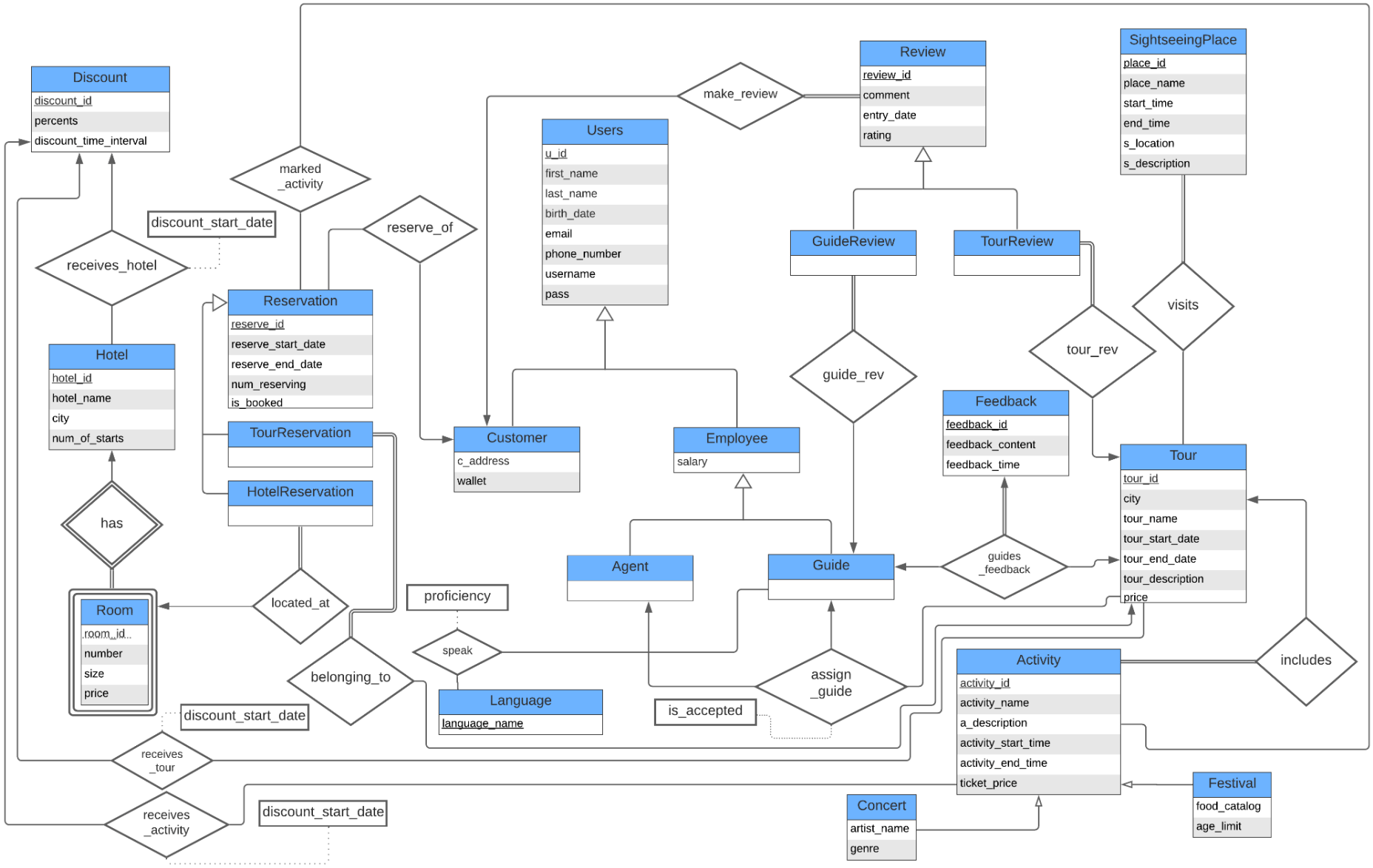

Discount
discount_id
percents
discount_time_interval
marked _activity
discount_start_date
receives_hotel
reserve_of
Reservation
reserve_id
reserve_start_date
reserve_end_date
num_reserving
is_booked
Hotel
hotel_id
hotel_name
city
num_of_starts
TourReservation
HotelReservation
has
Room
room_id
number
size
price
located_at
belonging_to
discount_start_date
receives _tour
receives _activity
discount_start_date
make_review
Review
review_id
comment
entry_date
rating
Users
u_id
first_name
last_name
birth_date
email
phone_number
username
pass
Customer
c_address
wallet
Employee
salary
GuideReview
TourReview
guide_rev
tour_rev
SightseeingPlace
place_id
place_name
start_time
end_time
s_location
s_description
visits
Feedback
feedback_id
feedback_content
feedback_time
Tour
tour_id
city
tour_name
tour_start_date
tour_end_date
tour_description
price
guides _feedback
Agent
Guide
proficiency
speak
Language
language_name
assign _guide
is_accepted
Activity
activity_id
activity_name
a_description
activity_start_time
activity_end_time
ticket_price
includes
Concert
artist_name
genre
Festival
food_catalog
age_limit

Figure 1: Final ER Diagram of the Travel Agency Web Application

3. Schemas

We have revised our Schemas based on the feedback given on the Design Report as follows:

- Representation of inheritance in Schemas was fixed, duplicate attributes were removed. Affected tables were the following:
    - Customer
    - Employee
    - Guide
    - Agent
    - GuideReview
    - TourReview
    - Festival
    - Concert
    - TourReservation
    - HotelReservation
- The primary key of guides_feedback was changed accordingly.
- The primary key of assign_guide was changed accordingly.
- Changes in E/R diagrams are applied accordingly.

3.1. Tour

Tour(tour_id, city, tour_name, tour_start_date, tour_end_date, tour_descritption, price, discount_id, discount_start_date)

PK: tour_id

FK: discount_id references Discount (if no discount is applied NULL)

3.2. Users

Users(u_id, first_name, last_name, birth_date, email, phone_number, username, pass)

PK: u_id

3.2.1. Customer

Customer(u_id, c_address, wallet)

PK: u_id

FK: u_id references Users

3.2.2. Employee

Employee(u_id , salary)

PK: u_id

FK: u_id references Users

3.2.2.1. Guide

Guide(u_id)

PK: u_id

FK: u_id references Employee

3.2.2.2. Agent

Agent(u_id)

PK: u_id

FK: u_id references Employee

3.3. Review

Review(review_id, comment, entry_date, rating, u_id)

PK: review_id

FK: u_id references Customer

3.3.1. GuideReview

GuideReview(review_id, u_id)

PK: review_id

FK: u_id references Guide

FK: review_id references Review

3.3.2. TourReview

TourReview(review_id, tour_id)

PK: review_id

FK: tour_id references Tour

FK: review_id references Review

3.4. Activity

Activity(activity_id, activity_name, a_description, activity_start_time, activity_end_time, ticket_price, tour_id, discount_id, discount_start_date)

PK: activity_id

FK: tour_id references Tour

FK: discount_id references Discount (if no discount is applied NULL)

3.4.1. Festival

Festival(activity_id, food_catalog, age_limit)

PK: activity_id

FK: activity_id references Activity

3.4.2. Concert

Concert(activity_id, artist_name, genre)

PK: activity_id

FK: activity_id references Activity

3.5. SightseeingPlace

SightseeingPlace(place_id, place_name, start_time, end_time, s_location, s_description)

PK: place_id

3.6. Hotel

Hotel(hotel_id, hotel_name, city, num_of_stars, discount_id, discount_start_date)

PK: hotel_id

FK: discount_id references Discount (if no discount is applied NULL)

3.7. Discount

Discount(discount_id, percents, discount_time_interval)

PK: discount_id

3.8. Reservation

Reservation(reserve_id, reserve_start_date, reserve_end_date, num_reserving, is_booked, u_id)

PK: reserve_id

FK: u_id references Customer

3.8.1. TourReservation

TourReservation(reserve_id, tour_id)

PK: reserve_id

FK: reserve_id references Reservation

FK: tour_id references Tour

3.8.2. HotelReservation

HotelReservation(reserve_id, room_id, hotel_id)

PK: reserve_id

FK: room_id, hotel_id references Room

3.9. Room

Room(room_id, hotel_id, number, size, price)

PK: room_id, hotel_id

FK: hotel_id references Hotel

3.10. Feedback

Feedback(feedback_id, content, feedback_time)

PK: feedback_id

3.11. Languages

Languages(language_name)

Primary key: language_name

3.12. visits

visits(place_id, tour_id)

Primary key: place_id, tour_id

FK: place_id references Sightseeing

FK: tour_id references Tour

3.13. speak

speak(u_id, language_name, proficiency)

Primary Key: u_id, language_name

FK: u_id references Guide

FK: language_name references Languages

3.14. guides_feedback

This is the relation between Guide, Tour, and Feedback, which represents how a Guide gives feedback at the end of the Tour.

guides_feedback(feedback_id, u_id, tour_id)

PK: feedback_id

FK: feedback_id references Feedback

FK: u_id references Guide

FK: tour_id references Tour

3.15. assign_guide

This is the relation between Agent, Tour, and Guide, which represents how an Agent assigns a Guide into a Tour.

assign_guide(guide_u_id, agent_u_id, tour_id, assign_status)

PK: tour_id

FK: tour_id references Tour

FK: agent_u_id references Agent (u_id)

FK: guide_u_id references Guide (u_id)

3.16. marked_activity

marked_activity(reserve_id, activity_id)

Primary Key: reserve_id, activity_id

FK: reserve_id references TourReservation

FK: activity_id references Activity

4. Implementation Details

In this project, we have used MSSQL Server as our database client. To execute queries on this database, we have used .NET Core 3.1 and to access this server-side through a webpage, we have used jQuery.

The first task was creating the database. For this, we have used Microsoft SQL Server Studio. There we have created our tables using CREATE TABLE queries. We also used SQL Server Studio for ALTER TABLE commands when a change was made in the ER Diagram.

To establish a local connection with the server, however, we have utilized Entity Framework Core, which generates classes out of an existing database, and creates other necessary classes, namely TravelAgencyContext.

To do this we have executed the following command, using a given connection string, in NuGet Package Manager Console:

```
Scaffold-DbContext "Data Source=(localdb)\MSSQLLocalDB;Initial
Catalog=TravelAgency;Integrated
Security=True;MultipleActiveResultSets=True;"
Microsoft.EntityFrameworkCore.SqlServer
```

As mentioned above we used a local server so the connection string was different for each member. To overcome this issue, we have included every team member's connection string in the necessary files. After that, we needed to set up the server side, which included adding the following: a CORS policy, TokenCheck filter for controllers, and a ResponseModel class.

After doing those changes, we were able to execute SQL queries on the server-side and send a response to the client-side. An example method is provided below how a basic query was executed:

```
/// <summary>
/// </summary>
/// <param name="userInfo"></param>
/// <returns></returns>
[HttpGet("action")]
0 references
public IActionResult Action()
{
    ResponseModel response = new ResponseModel();
    try
    {
        // SQL Queries here
        var tours = dbContext.Tours.FromSqlRaw("SELECT * FROM Tours").ToList();
        response.Data = tours;
    }
    catch (Exception ex)
    {
        // Catch SQL Exceptions, and send them to frontend
        response.HasError = true;
        response.ErrorMessage = ex.Message;
        if (ex.InnerException != null)
        {
            response.ErrorMessage += ": " + ex.InnerException.Message;
        }
    }
    // Return the HTTP response
    return Ok(response);
}
```

Figure 2: Example query execution

4.1. Problems Encountered

However, there were problems encountered. Those were basically:

- Lack of support for executing complex queries in .NET Core. In the code above, it can be observed that SELECT queries are executed using the **`FromSqlRaw`** method. However, the output type should match the table that was being accessed from, disabling the use of JOIN, GROUP BY, and so on.
- Setting up an MSSQL local server on Linux proved to be very challenging.
- Setting up SQL Server Management Studio on group members and synchronizing the data and configurations.

4.1.1. How Problems Were Solved

For the first problem, we have used the following function in Helper.cs to execute any SQL query.

```
public static List<T> RawSqlQuery<T>(string query, Func<DbDataReader, T> map)
{
    using (var context = new TravelAgencyContext())
    {
        using (var command = context.Database.GetDbConnection().CreateCommand())
        {
            command.CommandText = query;
            command.CommandType = CommandType.Text;

            context.Database.OpenConnection();

            using (var result = command.ExecuteReader())
            {
                var entities = new List<T>();

                while (result.Read())
                {
                    entities.Add(map(result));
                }

                return entities;
            }
        }
    }
}
```

Figure 3: Helper method used to execute any query

Below is the example usage of this function:

```
string finalQuery = "WITH results" +
    "(tour_id, city, tour_name, tour_start_date, tour_description, price, discount_id, tour_end_date) " +
    "AS (SELECT * " +
    "FROM Tour " +
    "WHERE " +
    dateQ +
    cityQ +
    priceQ +
    ") " +
    " SELECT tour_id, tour_name, city, tour_start_date, tour_end_date, tour_description, price, percents " +
    " FROM results LEFT JOIN Discount ON results.discount_id = Discount.discount_id; ";
Func<DbDataReader, TourDTO> map = x => new TourDTO
{
    tourId = (int)x[0],
    tourName = (string)x[1],
    city = (string)x[2],
    tourStartDate = (DateTime)x[3],
    tourEndDate = (DateTime)x[4],
    tourDescription = (string)x[5],
    price = (decimal)x[6],
    discountPercents = (x[7] != DBNull.Value)?((int)x[7]):0 // if percents is null, then the discount applied is zero percent
};
//var tours = dbContext.Tours.FromSqlRaw(finalQuery).ToList();
Console.Write(finalQuery);
var toursDTO = Helper.RawSqlQuery<TourDTO>(finalQuery, map);
// Get the discount of each tour
// Send an HTTP response as data, if necessary
response.Data = toursDTO;
```

Figure 4: Example usage for the helper method provided

4.2. Contribution of Group Members

Below is the detailed contribution of the group members.

4.2.1. Zübeyir Bodur

- Project setup. Includes activities such as importing ER diagrams into SQL Server database, building project architecture, updating database in case of a design change etc.
- Importing queries written by group members to Controller methods, so that they can be used in our system.
- Coming up with & implementing advanced database components
- Front-end work. Includes activities such as connecting Controller methods and the HTML pages, contributions in form validation, table representation, session control etc.
- Complex queries, such as reports (as advanced DB component), tour and hotel reservation history tables, and marked activity tables.

4.2.2. Abdul Razek Daher Katib

- Make a payment and its related queries.
- Add or update a review and a rating.
- List users and their reservations and info.
- UI design mock-ups.
- Building the models for implementation (later replaced by entity framework).
- MSSQL server setup on Linux.
- Testing website.

4.2.3. Emine Ezgi Saygılı

- Controller & queries for reviewing tour and guide
- Controller & queries for payment
- Controller & queries for listing all tours without an assigned guide
- Controller & queries for selecting a tour and assign an available guide
- Controller & queries for listing all available assigned tours

4.2.4. Funda Tan

- UI design and implementing frontend pages in HTML and CSS
- Importing queries to Controller methods that can be used in our system and connecting Controller methods with the HTML pages
- List all pending reservations query and show hotels query

5. Advanced Database Features

Below are the advanced database features used in our system.

5.1. Secondary Indices

- In the Users table, last_name is a secondary index. The last name of users may not be unique, however, we wanted to access the User's table using their last_name more efficiently. Hence, we used the SSMS client to create a nonclustered (secondary) index on the last_name column of this table.

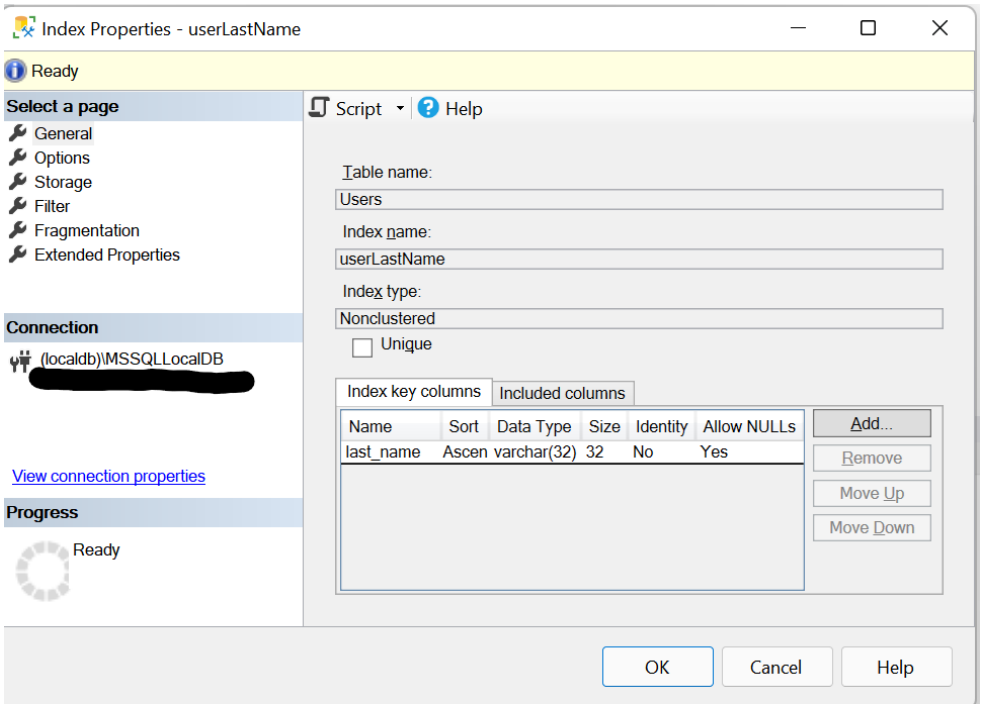


Figure 5: Preliminary (input) for the secondary index for last_name column, the output is not available

- In the Tours table, tour_name is a secondary index. The name of tours may not be unique, however, we wanted to access the User's table using their last_name more efficiently.

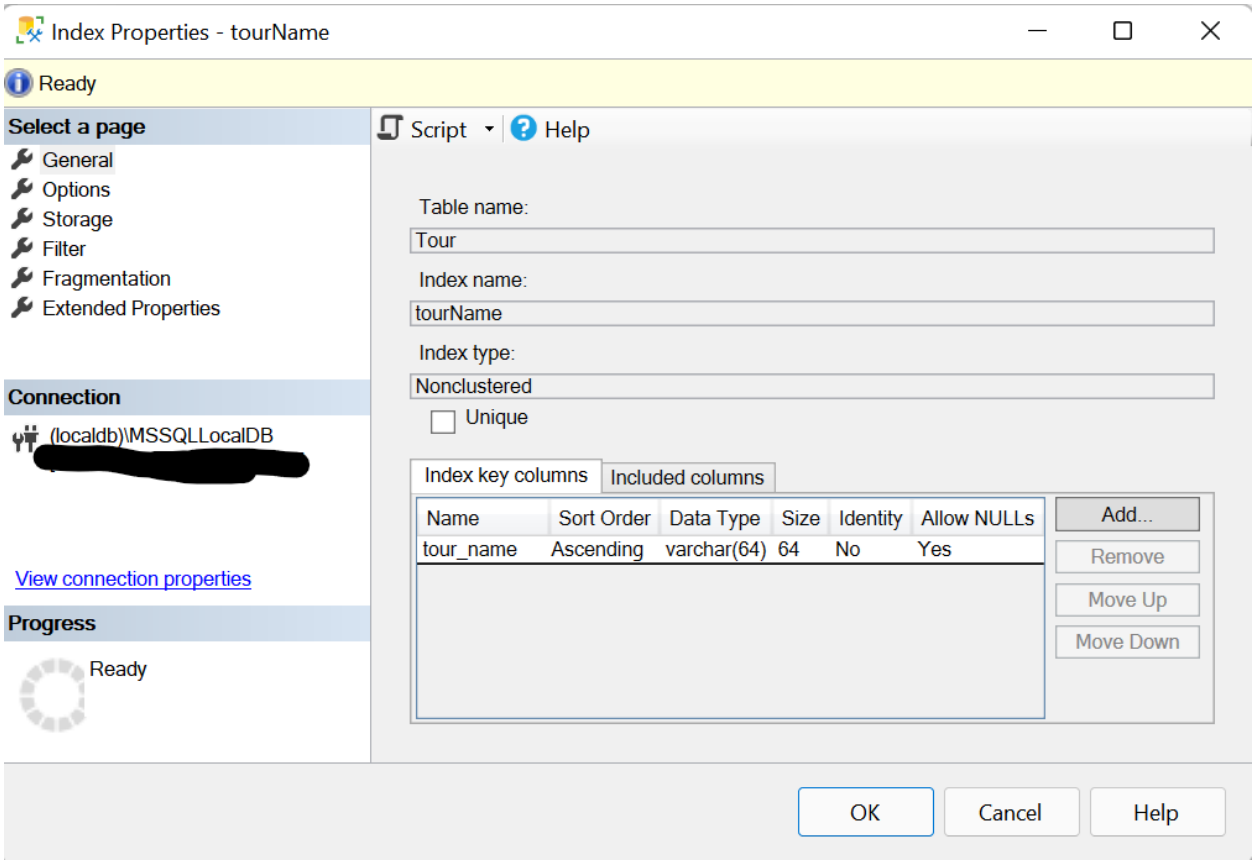


Figure 6: Preliminary (input) for the secondary index for tour_name column, the output is not available

5.2. Constraints

- In the tables that discounts can be applied (Tour, Hotel, and Activity), the mathematical expression

  `discount_id = NULL ⇔ discount_start_date = NULL`

  must be true.

  - In other words, if discount_id is NULL, then start_date must also be NULL. If discount_id is not NULL, then there must be a discount_start_date.

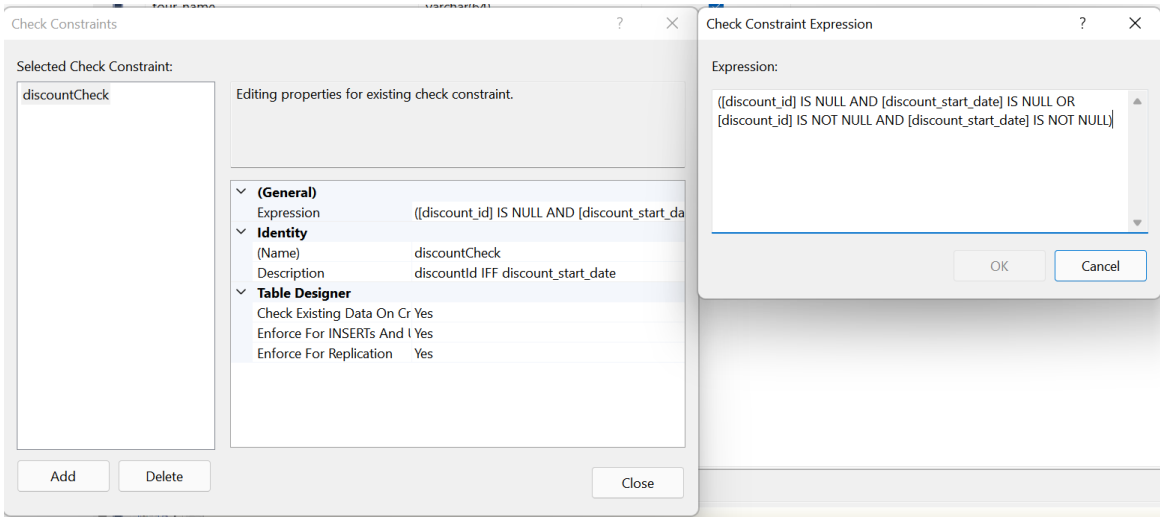


Figure 7: Preliminary (input) for the check constraint for discounts.

It can also be seen from the screen below, that this check constraint does not allow such entry of columns for discounts, as discount_id was NULL but discount_start_date was provided:

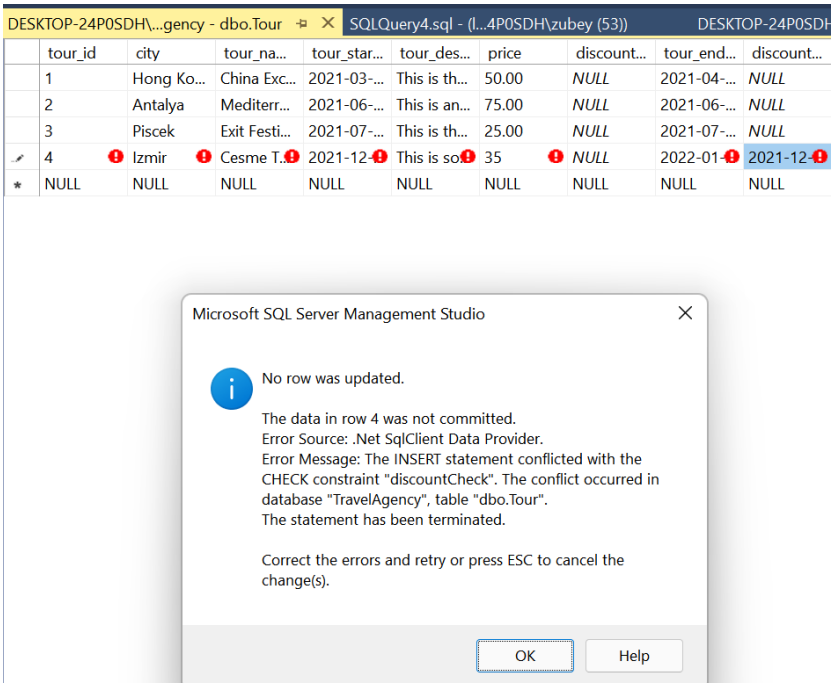


Figure 8: Preliminary (output) for the check constraint for discounts.

5.3. Reports

5.3.1. Retrieve the payment history for tour reservations and find the total number of money spent for each customer for tour reservations

The query below will retrieve the payment history for tour reservations. This query can also be used to find out the total spendings of each customer. It should be noted that the spendings of a customer will be zero for a reservation that has not been booked yet.

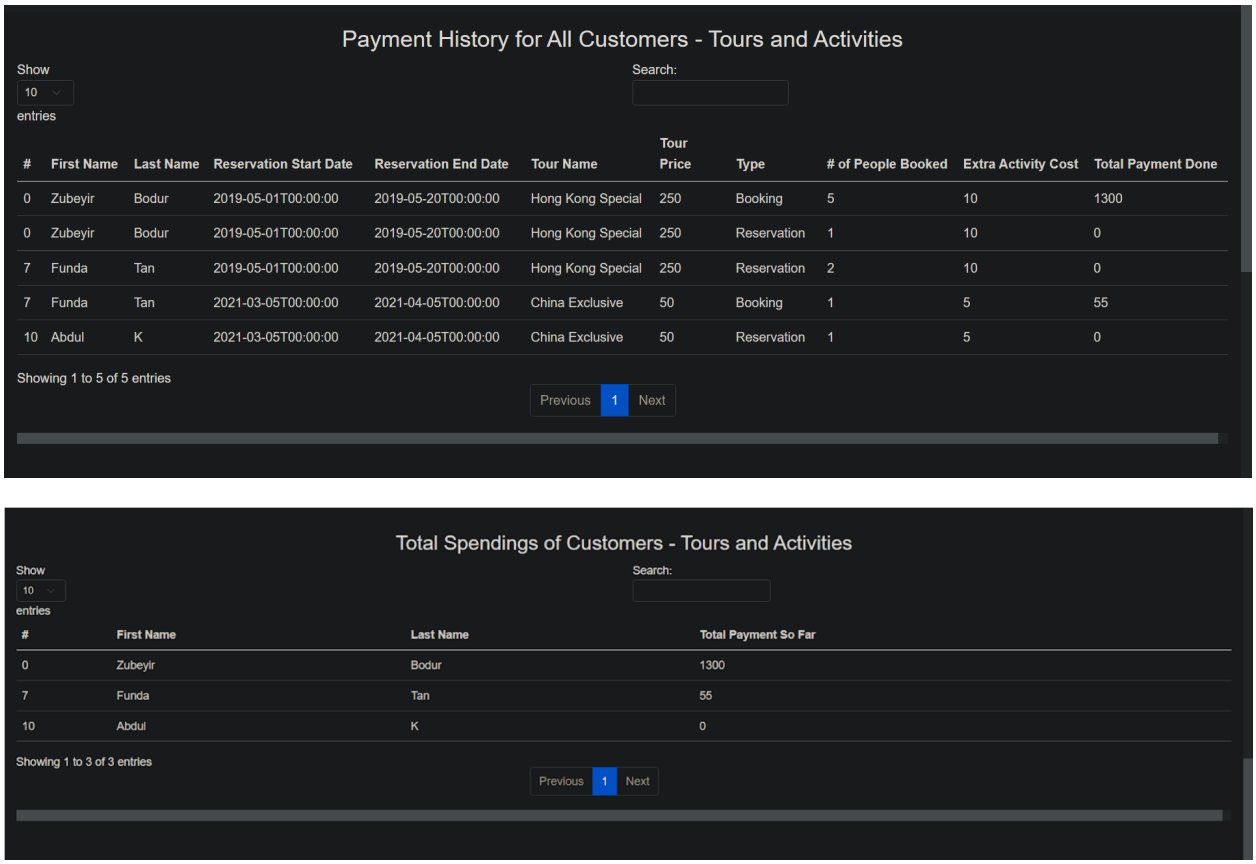
```
WITH paymentHistory(reserve_id, reserve_start_date, reserve_end_date,
                        u_id, first_name, last_name, tour_id,
                        tour_name, price, is_booked,
                        num_reserving,
                        total_ticket_price_per) AS
    (SELECT Reservation.reserve_id, reserve_start_date,
                reserve_end_date, Users.u_id,
                Users.first_name, Users.last_name,
                Tour.tour_id, Tour.tour_name,
                price, is_booked, num_reserving,
                SUM(ticket_price) as total_ticket_price_per
    FROM Reservation    JOIN TourReservation ON Reservation.reserve_id = TourReservation.reserve_id
                        JOIN Tour ON Tour.tour_id = TourReservation.tour_id
                        JOIN marked_activity ON Reservation.reserve_id = marked_activity.reserve_id
                        JOIN Activity ON Activity.activity_id = marked_activity.activity_id
                        JOIN Users ON Reservation.u_id = Users.u_id
    GROUP BY Reservation.reserve_id, reserve_start_date,
                reserve_end_date, Users.u_id,
                Users.first_name, Users.last_name,
                Tour.tour_id, Tour.tour_name,
                price, is_booked,
                num_reserving)
SELECT  u_id,
        first_name,
        last_name,
        reserve_start_date,
        reserve_end_date,
        tour_name,
        price,
        is_booked,
        num_reserving,
        total_ticket_price_per as extra_activities_per,
        is_booked*num_reserving*(price + total_ticket_price_per) as total_payment_done
FROM paymentHistory
ORDER BY last_name ASC;
```

Figure 9: The query that will retrieve the payment history for tour reservations

```
WITH paymentHistory(reserve_id, reserve_start_date, reserve_end_date,
                        u_id, first_name, last_name, tour_id,
                        price, is_booked,
                        num_reserving, total_ticket_price_per) AS
    (SELECT Reservation.reserve_id, reserve_start_date,
                reserve_end_date, Users.u_id, Users.first_name,
                Users.last_name, Tour.tour_id, price,
                is_booked, num_reserving,
                SUM(ticket_price) as total_ticket_price_per
    FROM Reservation    JOIN TourReservation ON Reservation.reserve_id = TourReservation.reserve_id
                        JOIN Tour ON Tour.tour_id = TourReservation.tour_id
                        JOIN marked_activity ON Reservation.reserve_id = marked_activity.reserve_id
                        JOIN Activity ON Activity.activity_id = marked_activity.activity_id
                        JOIN Users ON Reservation.u_id = Users.u_id
    GROUP BY Reservation.reserve_id,
                reserve_start_date,
                reserve_end_date,
                Users.u_id,
                Users.first_name,
                Users.last_name,
                Tour.tour_id,
                price,
                is_booked,
                num_reserving)
SELECT u_id, first_name, last_name,SUM(total_payment_done) AS total_payment_so_far
FROM (
    SELECT  u_id, first_name, last_name,
            price, is_booked, num_reserving, total_ticket_price_per,
            is_booked*num_reserving*(price + total_ticket_price_per) as total_payment_done
    FROM paymentHistory) AS paymentHistoryFinal
GROUP BY u_id, first_name, last_name
ORDER BY total_payment_so_far DESC
```

Figure 10: The query that will retrieve total payment done by a single customer in tour reservations

5.3.1.1. Output screen



Payment History for All Customers - Tours and Activities

Show 10 entries

Search:

#	First Name	Last Name	Reservation Start Date	Reservation End Date	Tour Name	Tour Price	Type	# of People Booked	Extra Activity Cost	Total Payment Done
0	Zubeyir	Bodur	2019-05-01T00:00:00	2019-05-20T00:00:00	Hong Kong Special	250	Booking	5	10	1300
0	Zubeyir	Bodur	2019-05-01T00:00:00	2019-05-20T00:00:00	Hong Kong Special	250	Reservation	1	10	0
7	Funda	Tan	2019-05-01T00:00:00	2019-05-20T00:00:00	Hong Kong Special	250	Reservation	2	10	0
7	Funda	Tan	2021-03-05T00:00:00	2021-04-05T00:00:00	China Exclusive	50	Booking	1	5	55
10	Abdul	K	2021-03-05T00:00:00	2021-04-05T00:00:00	China Exclusive	50	Reservation	1	5	0

Showing 1 to 5 of 5 entries

Previous 1 Next

Total Spendings of Customers - Tours and Activities

Show 10 entries

Search:

#	First Name	Last Name	Total Payment So Far
0	Zubeyir	Bodur	1300
7	Funda	Tan	55
10	Abdul	K	0

Showing 1 to 3 of 3 entries

Previous 1 Next

Figure 11: The preliminaries for payment histories of customers - tour reservations only

5.3.2. Display the top 10 performing guides according to the reviews they got

The query below will display the first name, last name of the guides that were chosen best by their customers, according to the reviews they got.

```
WITH guide_ratio(u_id, avg_stars) AS(
    SELECT Guide.u_id, AVG(Review.rating) as avg_stars
    FROM Guide  JOIN GuideReview ON Guide.u_id = GuideReview.u_id
                JOIN Review ON Review.review_id = GuideReview.review_id
    GROUP BY Guide.u_id)
SELECT TOP 10 guide_ratio.u_id, first_name, last_name, avg_stars
FROM guide_ratio JOIN Users ON guide_ratio.u_id = Users.u_id
ORDER BY avg_stars DESC
```

Figure 12: The query that will display the top 10 guides

5.3.2.1. Output screen

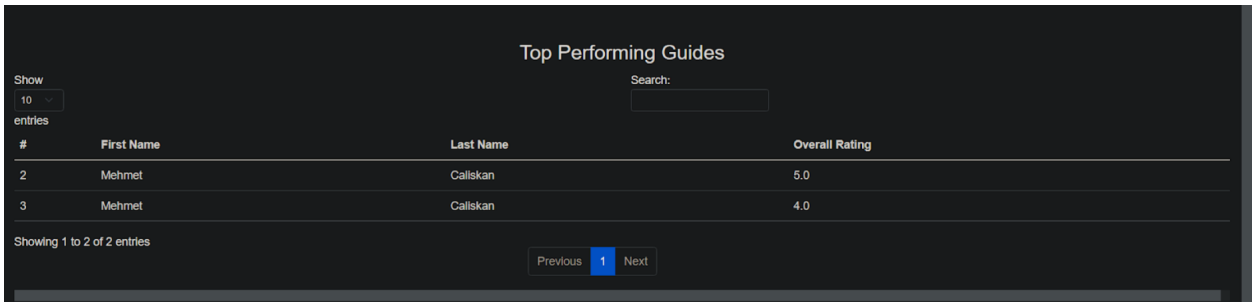


Figure 13: The preliminary (output) for displaying the top 10 guides

6. User's Manual

Below is the user manual of the system, described with actual mock ups from the website.

6.1. Login and Sign-up Pages

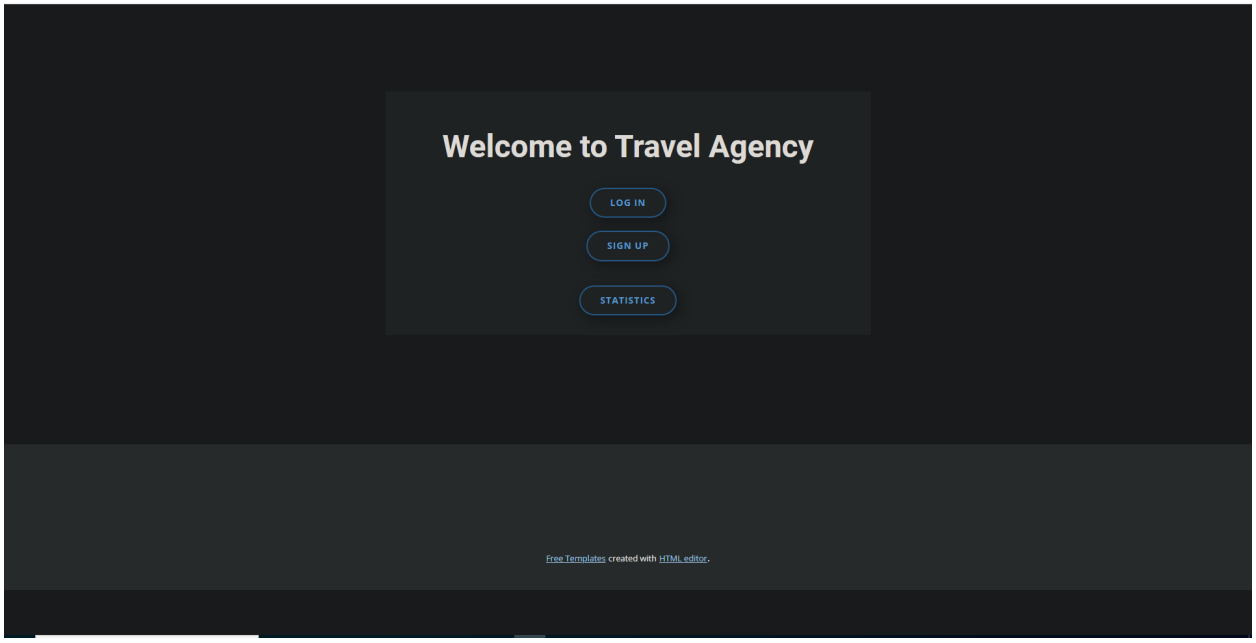


Figure 14: Index page

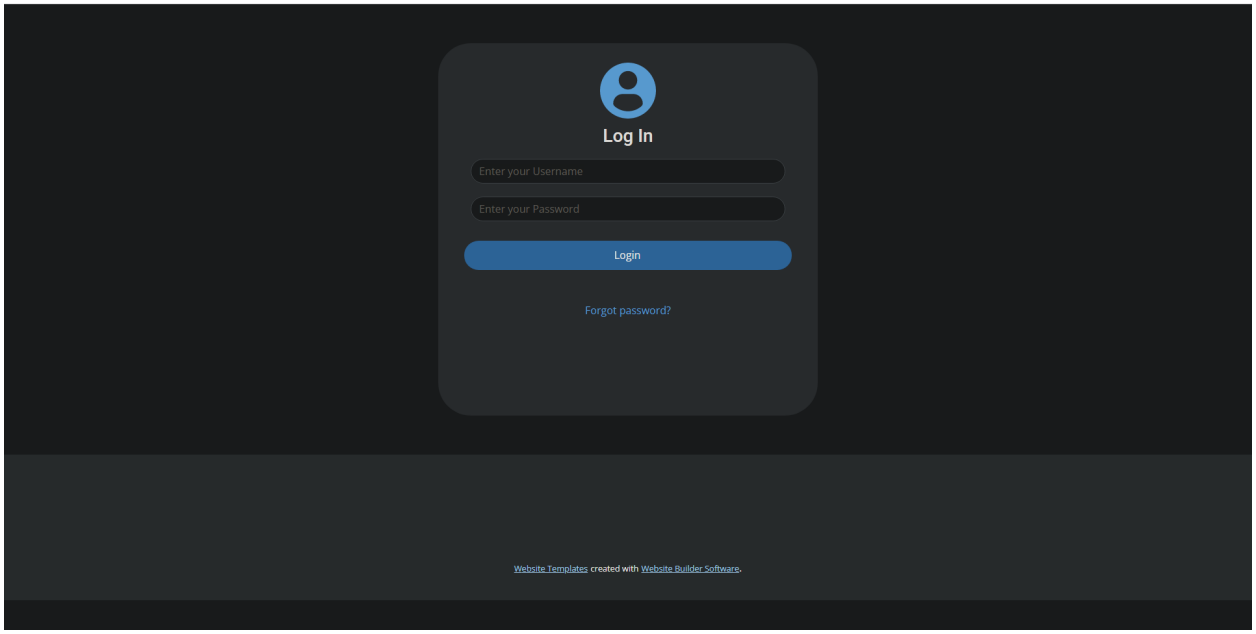


Figure 15: Login page

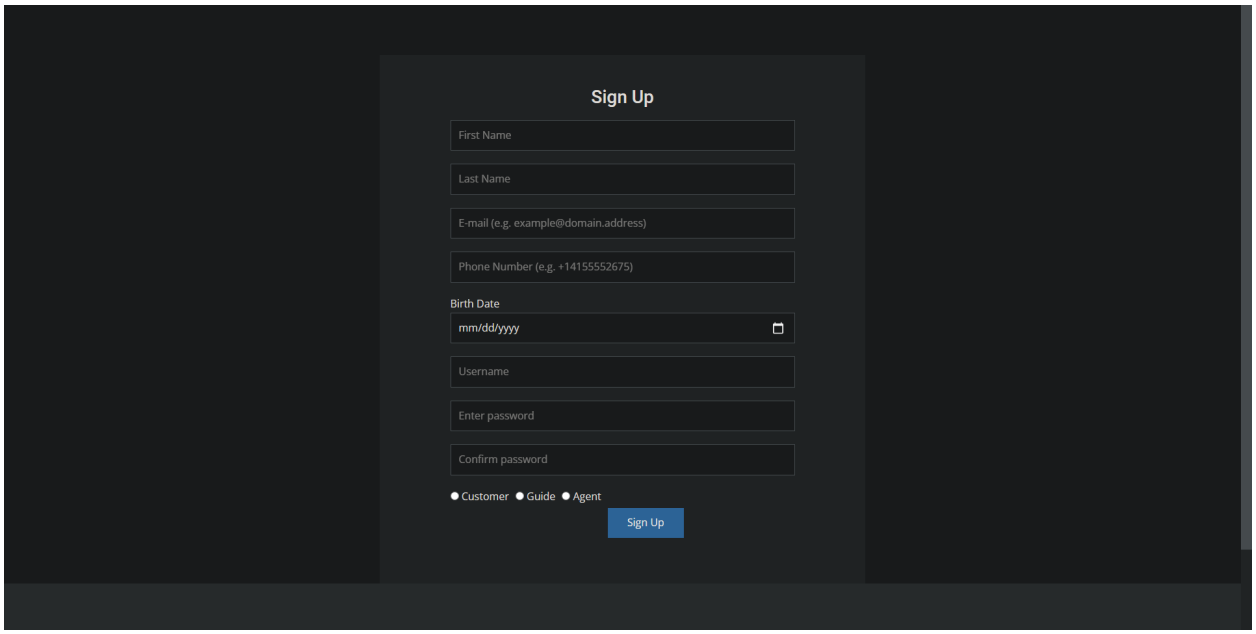


Figure 16: Sign-up page

The index page of the system users will be provided with three buttons to take them to the next page: login, sign up, or statistics.

Login page prompts users to enter a username and a password or to click “Forgot Password?” in case they forgot their passwords. In case the user has no account they can go to the Sign-Up

page and create an account, afterwards they will be redirected to the main page according to the user type.

In the Sign-Up Page, there is various information that has to be provided. These are first name, last name, e-mail, phone number, birth date, password and confirm password. Users have to provide these to sign-up. Then, they need to select their user type from types, customer, guide and agent. When the sign-up process is over, they are redirected to the Login Page. The statistics page will be reviewed in the next section.

6.2. Statistics Page

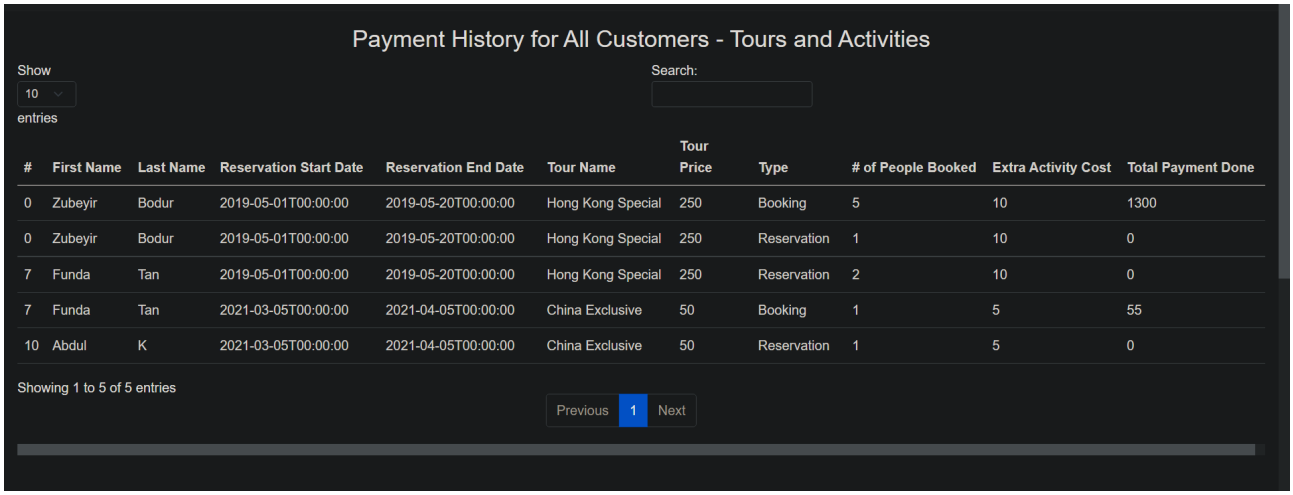


Figure 17: Statistics page, Payment History section

The statistics consists of three sections that show a brief analysis of the data in the system. In the first section a table of payments history in the system is shown.

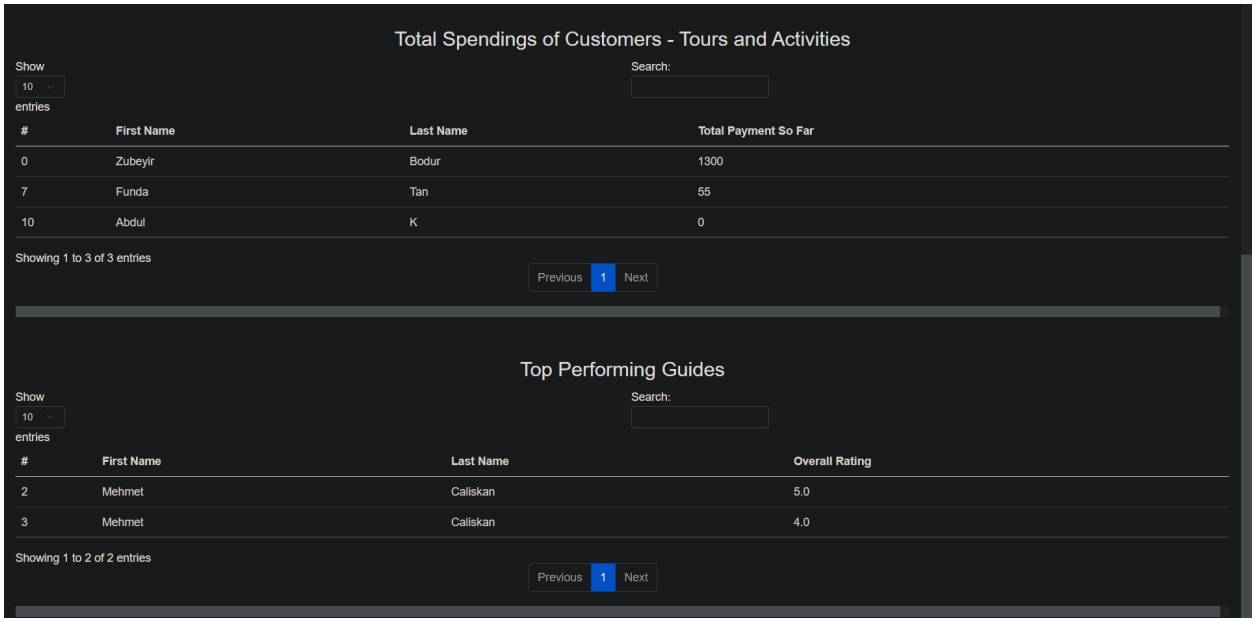


Figure 18: Statistics page, Total Spendings and Top Performing Guides sections

The next two sections are the total spendings of each customer and a list of top performing guides based on their ratings.

6.3. Customer Page

6.3.1. Reserving and Booking

6.3.1.1. Listing Available Tours

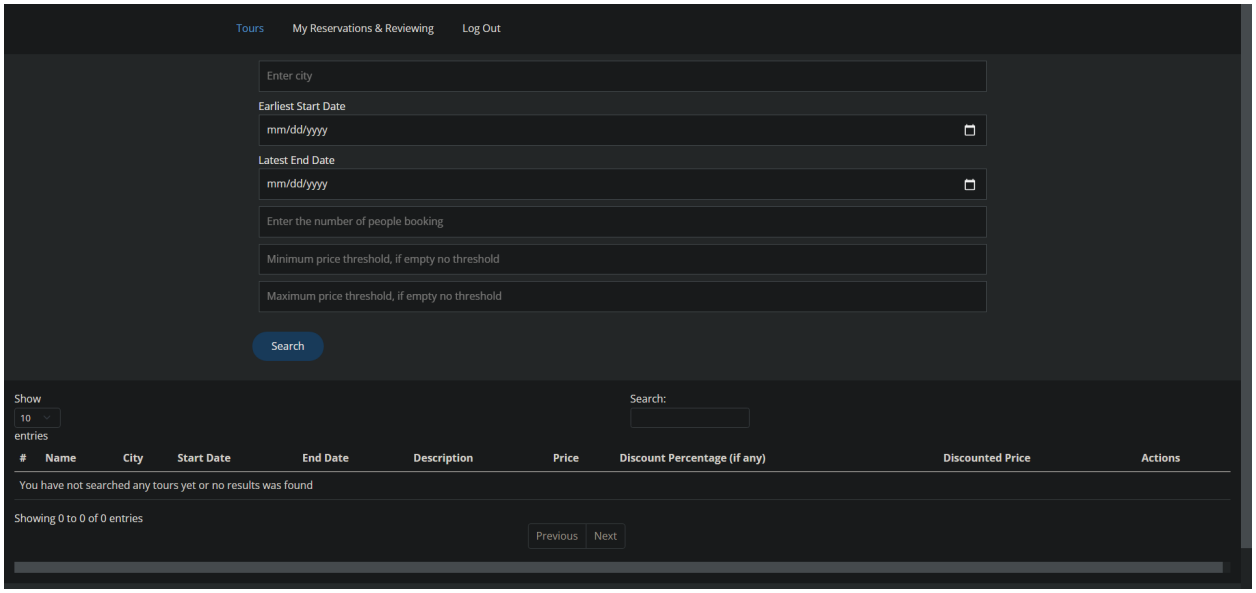


Figure 19: Main page of customer.

The main page after the user logs in is a page with a search section, with the topbar providing the “My reservations and Booking” page and the “Logout” choice.

6.3.1.2. Selecting a Tour

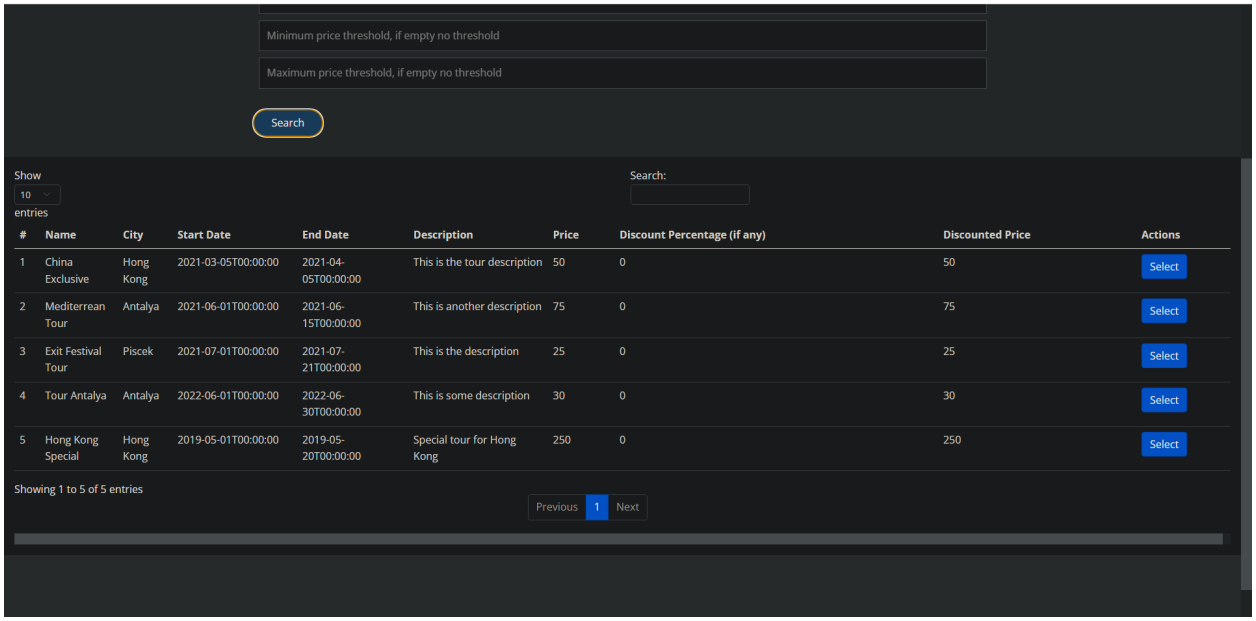


Figure 20: Search results on the main page.

From the search bar, users can make a search by using entries. Also, they can choose how many items will be presented in the Results page and switch between the pages using Previous and Next buttons for all three tables. Users can enter a city name and choose the earliest and latest start dates. Search is flexible so that the user is only required to enter the number of people booking to get a list of destinations to visit. Then, they can provide a number of people for booking, minimum and maximum thresholds for the tour to be used as a filter. When they press the search button, related tours will be presented with its information. When users select a tour, they are sent to the Activities and Hotels page.

6.3.1.3. Choosing Extra Activities to Attend & Hotel Reservation

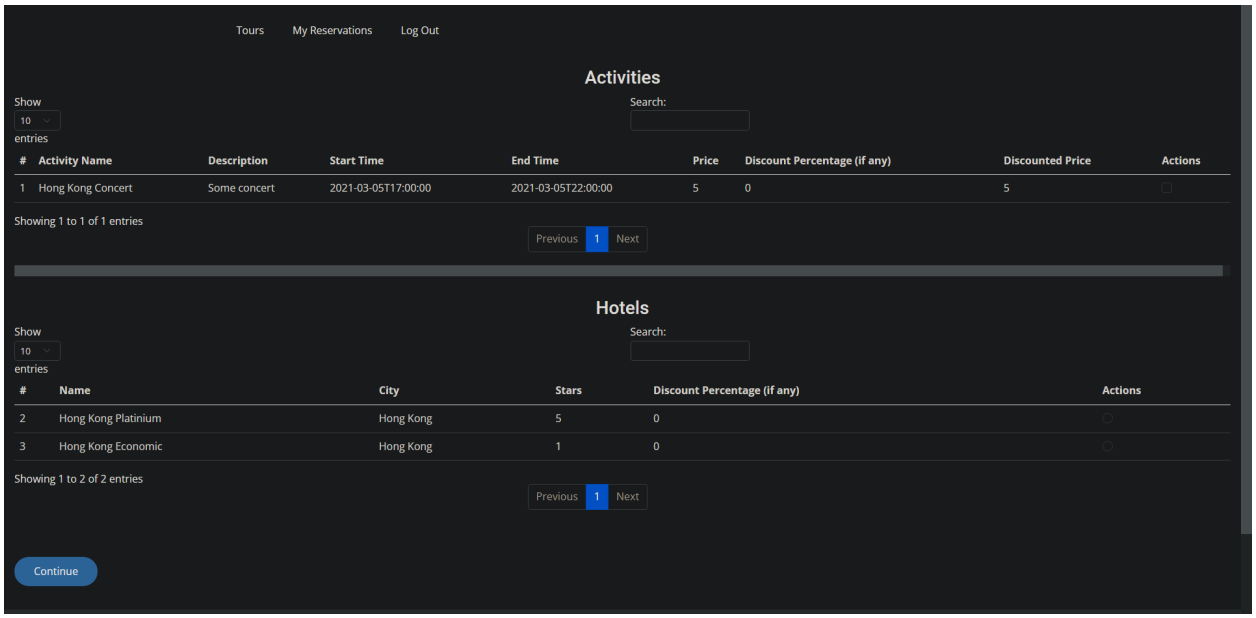


Figure 21: Activities and Hotels page.

In the Activities and Hotels page, users can select activities for their tour and they can select a hotel, along with the number of stars, to stay at. When they press the Continue button they will be sent to the Rooms page of the selected hotel.

6.3.1.4. Room Selection for Hotel Reservation

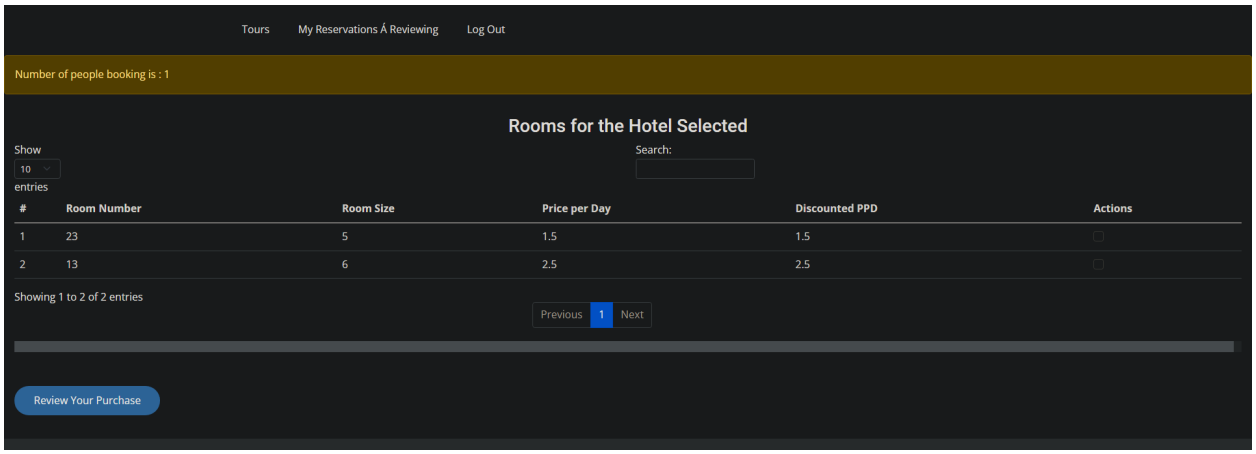


Figure 22: Available rooms in hotel page.

In this page, users can see rooms and select one, or more, using check boxes. When the “Review Your Purchase” button is clicked, the user is sent to the Payment Summary Page to make the payment.

6.3.1.5. Payment Summary

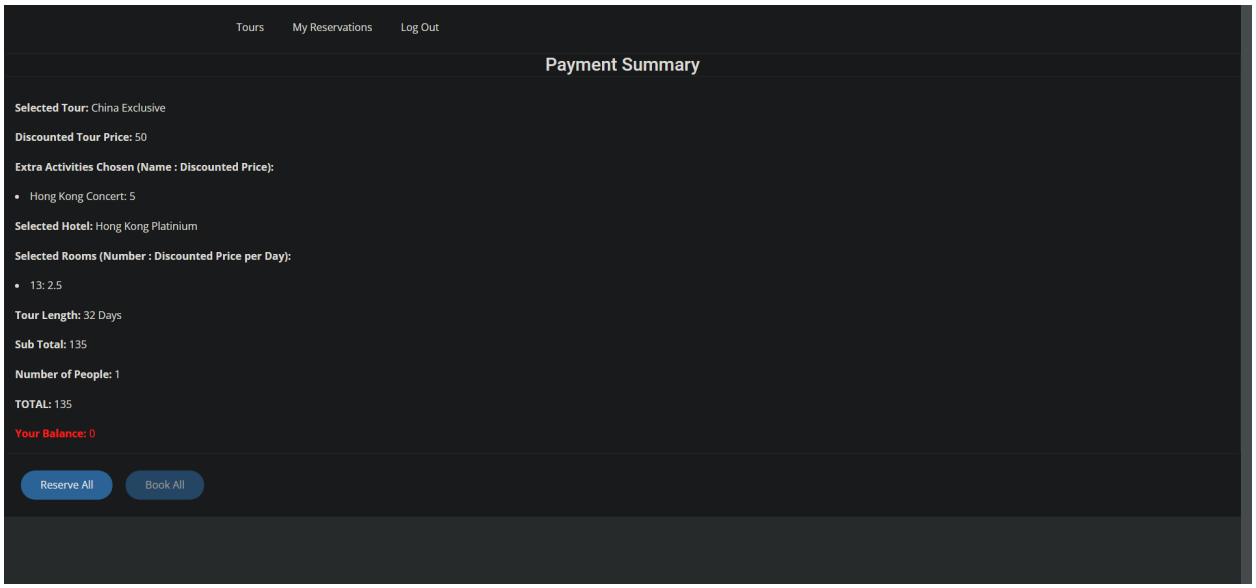


Figure 23: Payment summary page.

In the Payment Summary page, users see the summary of the payment, which shows the amount to be paid, discount included, and other details related to their payment. They can finish the process by pressing Reserve All or Book All buttons. If the user presses Book All, they will be performing booking and hence making a payment. Otherwise, they will be making reservations for the tour and hotel they have chosen.

6.3.2. My Reservations and Reviewing Page

6.3.2.1. Displaying Customer's Reservations

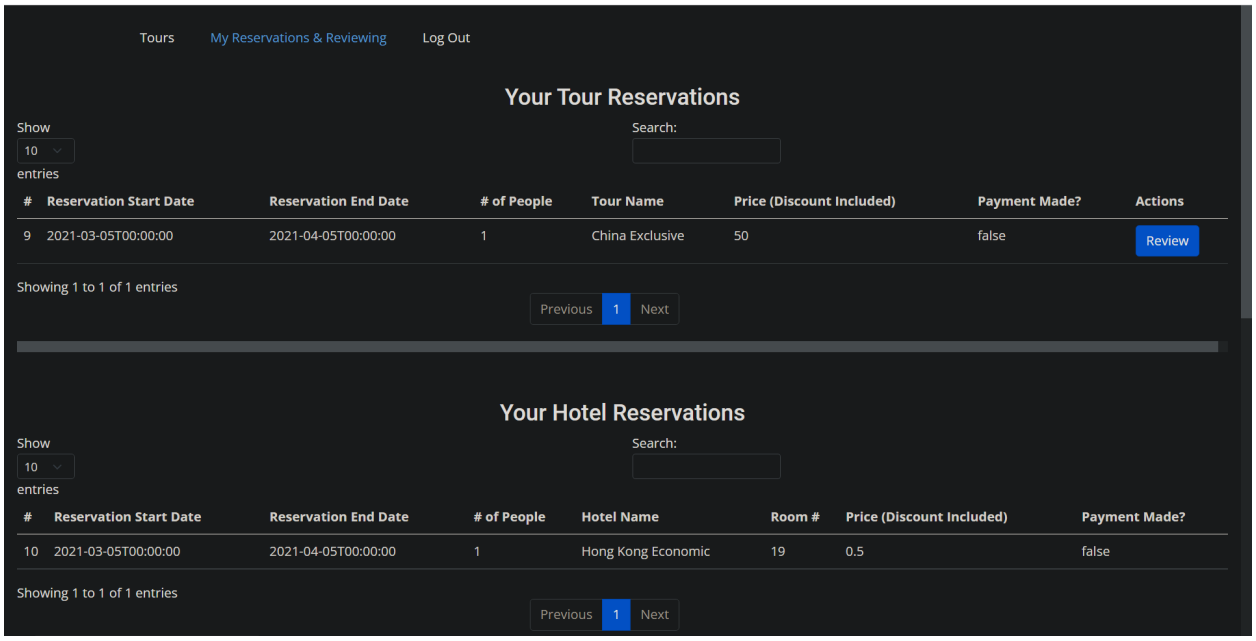


Figure 24: Reservations and reviewing page.

After a user makes a reservation for a tour they can see that tour in the "My Reservations and Reviewing" Page, they can reach this page by clicking on the option in the topbar. From there they can see the tours they reserved in the top section, and their hotel reservation in the bottom section. Users can click on the "Review" button to be taken to the next page where they can provide a rating and write a review for the tour and guide as will be shown in the next section.

6.3.2.2. Reviewing a Tour and Its Guide

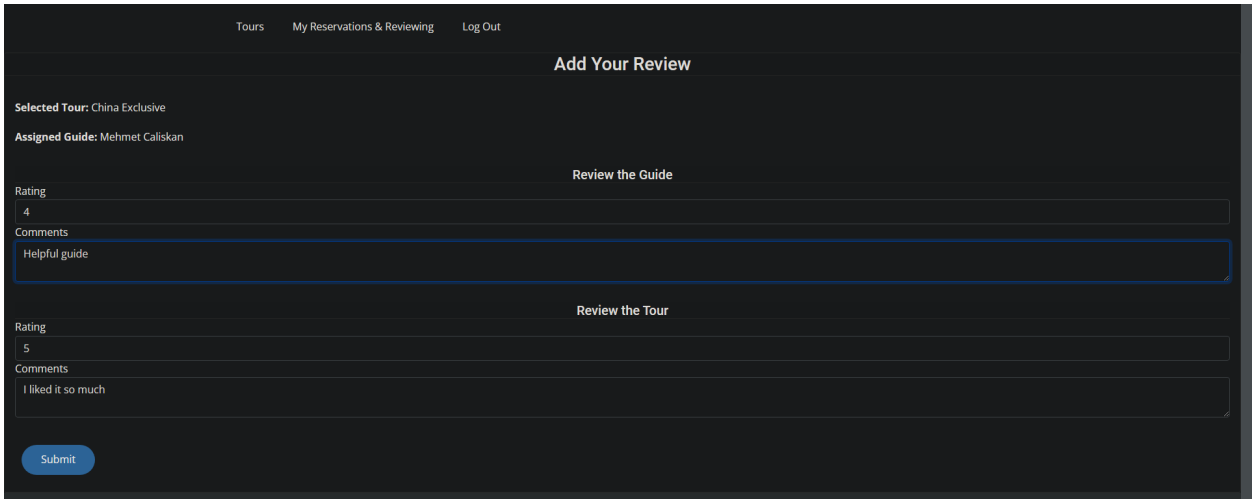


Figure 25: Reservations and reviewing page.

In this page users are provided with the name of the selected tour and its guide. The user can rate each by giving each point out of 5. Then they can proceed to write a review. Then once they are done they can click on “Submit” to submit the reviews and rating to the system. Note that the review field is optional while the rating field is mandatory.

6.4. Agent Page

6.4.1. Tours without an Assigned Guide & Assigning a Guide

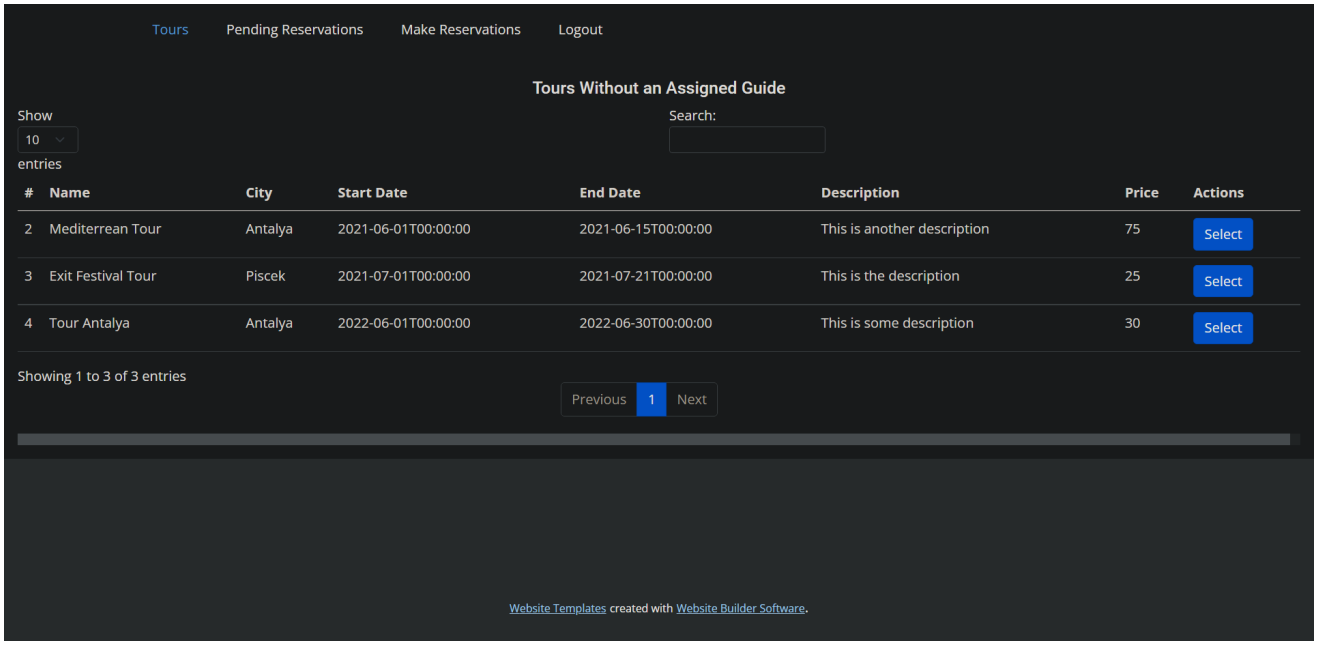


Figure 26: List tours without a guide page.

An agent will be provided with a different page upon their login. The provided page shows a list of the tours that do not have an assigned guide yet. The agent can choose each tour and then assign a guide.

6.4.2. Make Reservations for a Customer

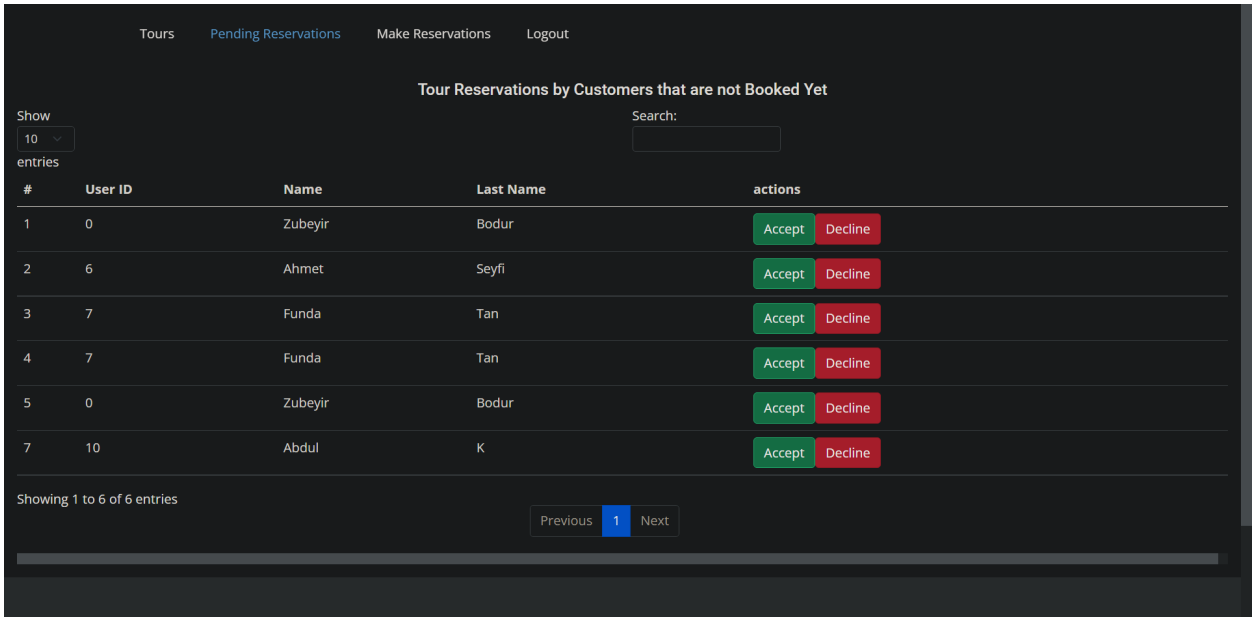


Figure 27: Pending reservations page

When the agent clicks on the “Pending Reservations” option from the topbar, they are taken to a page with reservations that were made by users but were not booked, i.e no payments were made. They can then accept or decline each reservation.

6.4.3. Make Reservations for a Customer

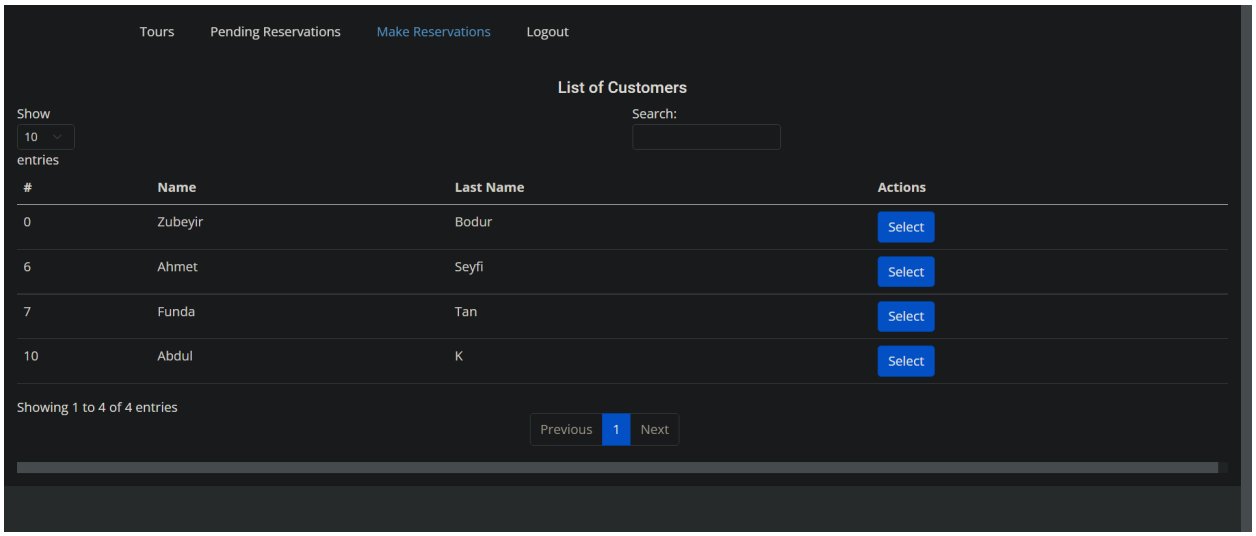


Figure 28: Reserved tours management page.

When the agent clicks on the “Make Reservations” option from the topbar, the system takes them to a page with the list of all customers in the system. The agent, afterwards, can select a user then assign a tour to them. The purpose of this page is to give the agents the ability to reserve for customers in case the customers chose not to do it themselves.

6.5. Guide Page

Since this page was not implemented, when a Guide logs in to the system, they will be shown an error page, and then be redirected to the Index page after automatically being logged out.

7. References

[1] “Travel Agency Management System | https://cs353-travel-agency-system.github.io,” *GitHub Pages,* [Online]. Available : https://cs353-travel-agency-system.github.io/. [Accessed Jan. 4, 2022].