Montreal, March 12th 2023

FSD07 - Web Services - Team Project

Team: Bastos, Debora
Dhakal, Arjun
Sawyer, Mackenzie

Note: The project we are presenting is based on the instructions outlined in the document appended at the end of this report.

---

1. Explanation of the Project

---

a) Domain Description:

The ABC Restaurant domain provides information on the restaurant including an about section, its location, contact information, business hours, menu/specials, customer reviews and external links (social media pages). The domain features several pages including a home, menu, about us, contact and login page. On the home page there is a “Book A Table” button for users to book a reservation. The ABC Restaurant domain also includes a login page where employees can login to view and manage reservations. Future implementations to the domain would include a login function for customers, where once registered, customers are able to place orders for takeout/delivery.

b) Web API Description:

Reservation Form

- The reservation form collects the date & time the reservation was made and the number of guests. The form also collects the user's first and last name, email, phone number and includes a field for special requests (highchair for infants, disability accommodation, etc.). The form collects the table location (table or bar) and the number of guests.

- The reservation form delivers the client request to the database where it checks if there are tables with the amount of guests available at the date and time requested. If not, the user will be prompted to try another time, or if the day is fully booked, to try another date.

- If a reservation is successfully sent to the server, the customer receives a confirmation email with a reservation id (confirmation) number.

c) List of Routes

Reservation Controller (API routes for reservation, where “id” = reservation id)

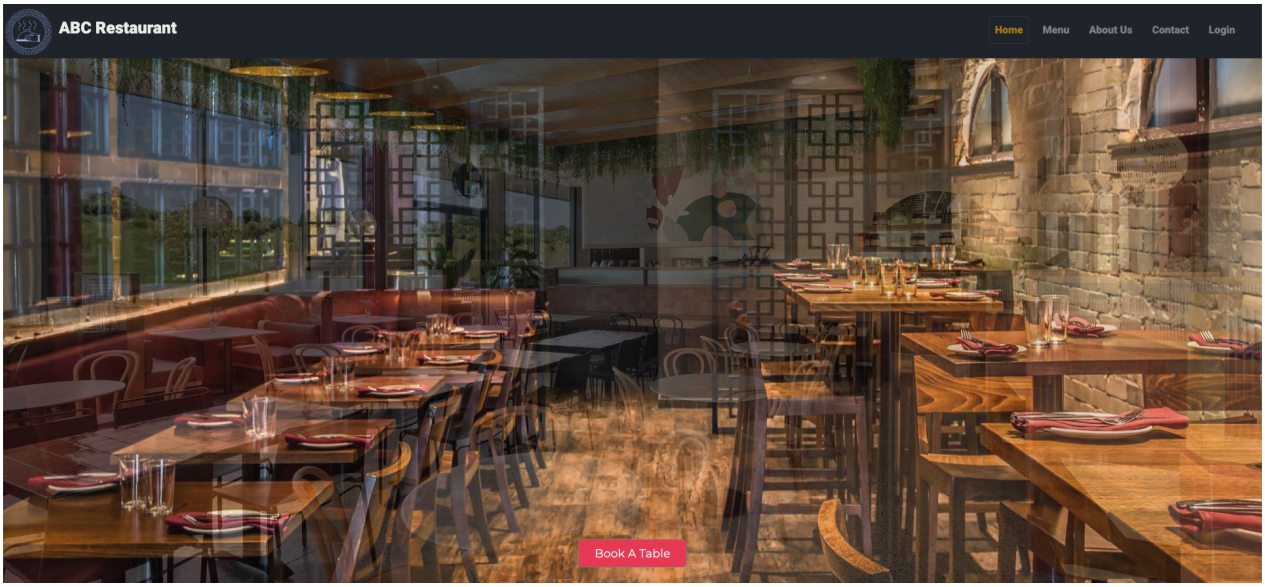
- GET (“/admin/reservation/all”)
    - Get all reservations
- GET (“/admin/reservation/date”)
    - Get reservation(s) by date
- GET (“/admin/reservation/id”)
    - Get reservation by id
- GET (“/admin/reservation/email”)
    - Get reservation by customer email
- GET (“/admin/reservation/name”)
    - Get reservation by customer name (can search by first or last name or any combination of letters)
- POST (“/reservation-form”)
    - Create a new reservation
- PUT ("/admin/reservation/update")
    - Update a reservation
    - Search by reservation id
- PUT ("/admin/reservation/cancel")
    - Cancel an existing reservation by searching the reservation id
- GET ("/reservation/tables")
    - Get available tables
    - Search by date and time

Employee Controller (API routes for employee, where “id” = employee id)
@RequestMapping(“/abc”) is used.

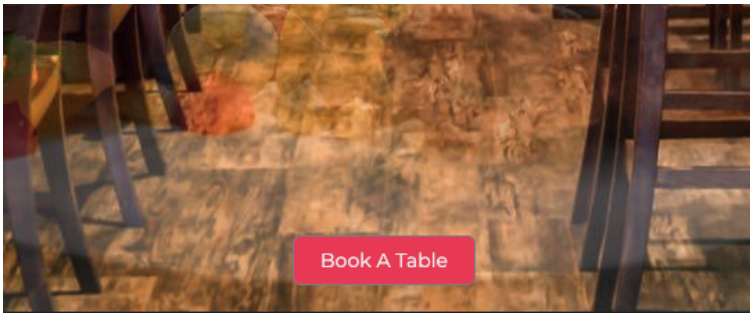
- GET (“/employees”)
    - Get all employees
- GET (“/employees/{employeeId}”)
    - Get employee by id
- GET ("/employees/employee")
    - Get employee by employee’s email
- POST (“/employees”)
    - Create a new employee
- PUT ("/employees/{id}")
    - Update an existing employee

2. MOCKUP of the UI

Domain Homepage



Description:
Tabs located in the top right corner link to the websites different pages.

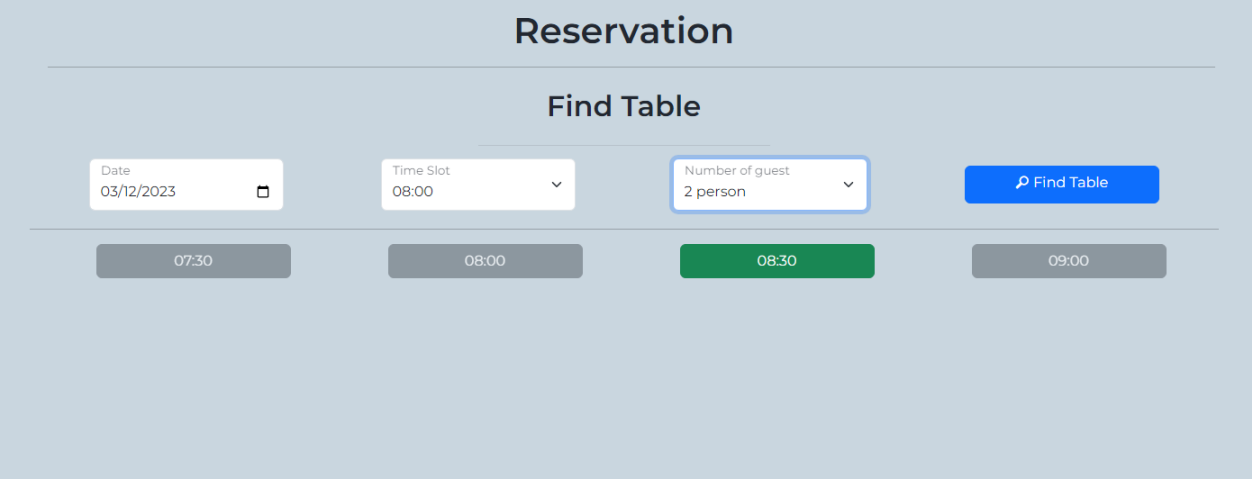


Description:
“Book A Table” button directs the user to the Reservation page form.

Reservation Page (for Customers)

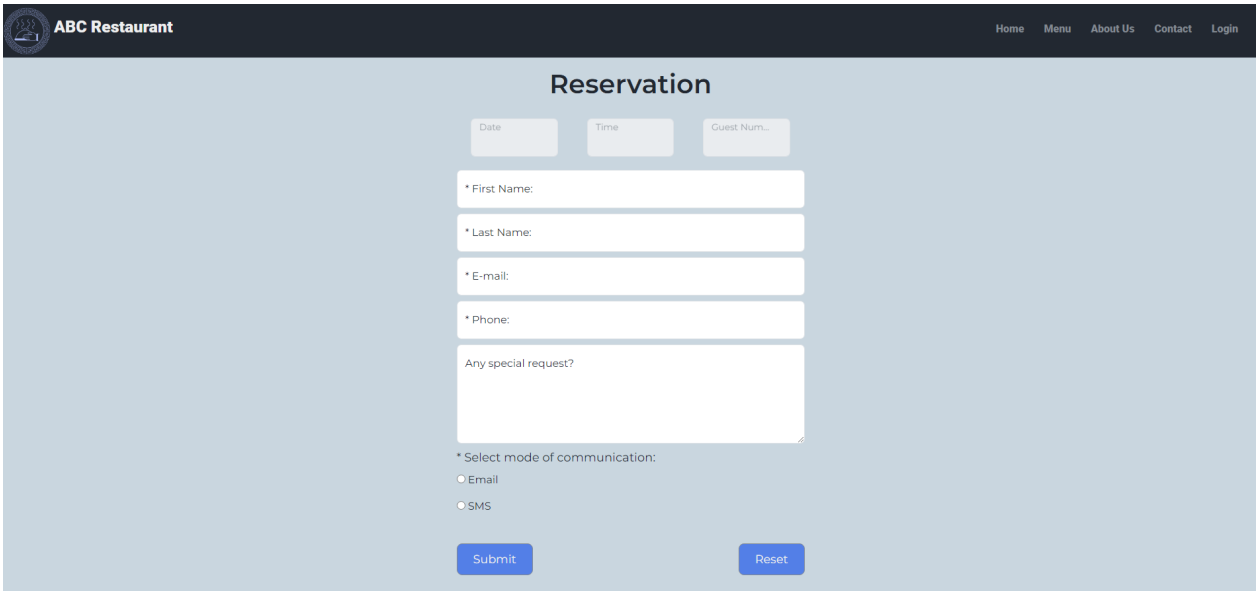
First Page - Find Table

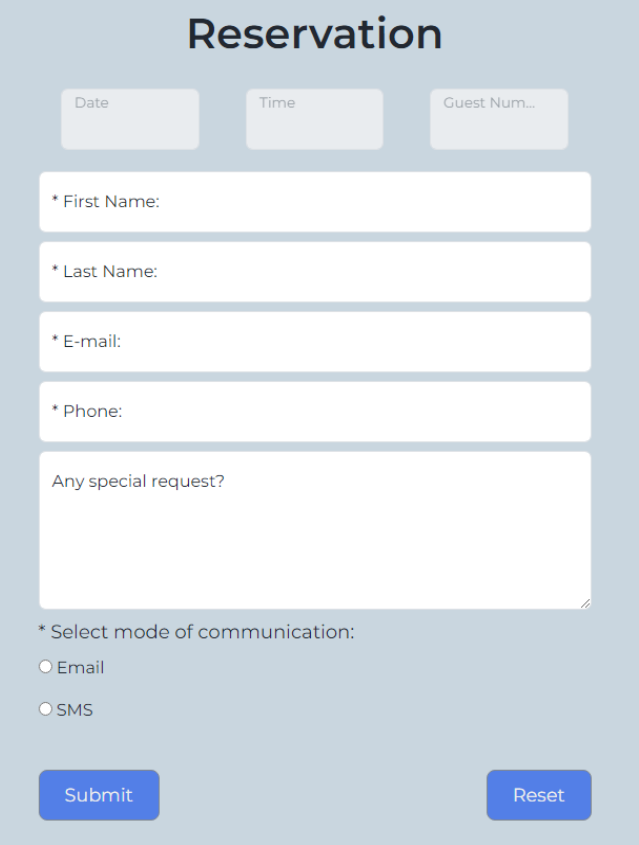
Customer searches table availability by entering the date and number of guests. The website (front end) is connected by the database (back end) and shows up to date time slots available in green.



Second Page - Reservation Form

Customers enter information and confirm their reservation by clicking the submit button.





Description:

The reservation page uses a form to collect data from customers to book their table.

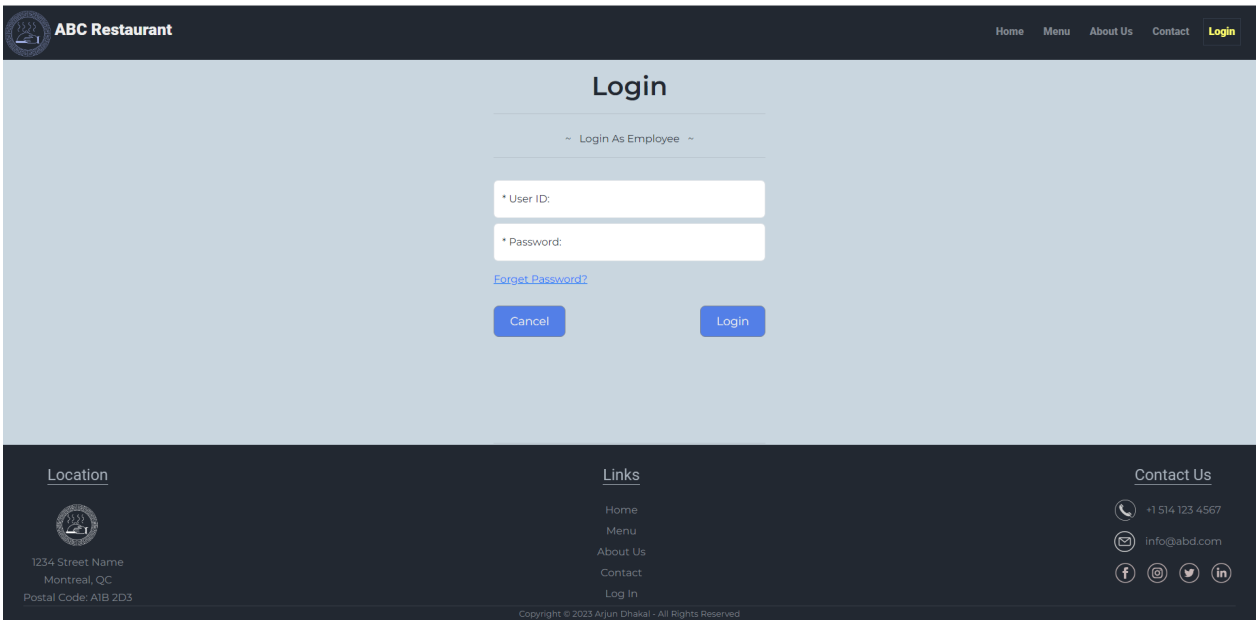
Required fields:

- First Name
- Last Name
- E-mail
- Phone
- Mode of communication

The optional text box stores any special requests from the customer (disability accommodation, highchairs for children, allergies, etc.).

Reservation Page (for Employees)

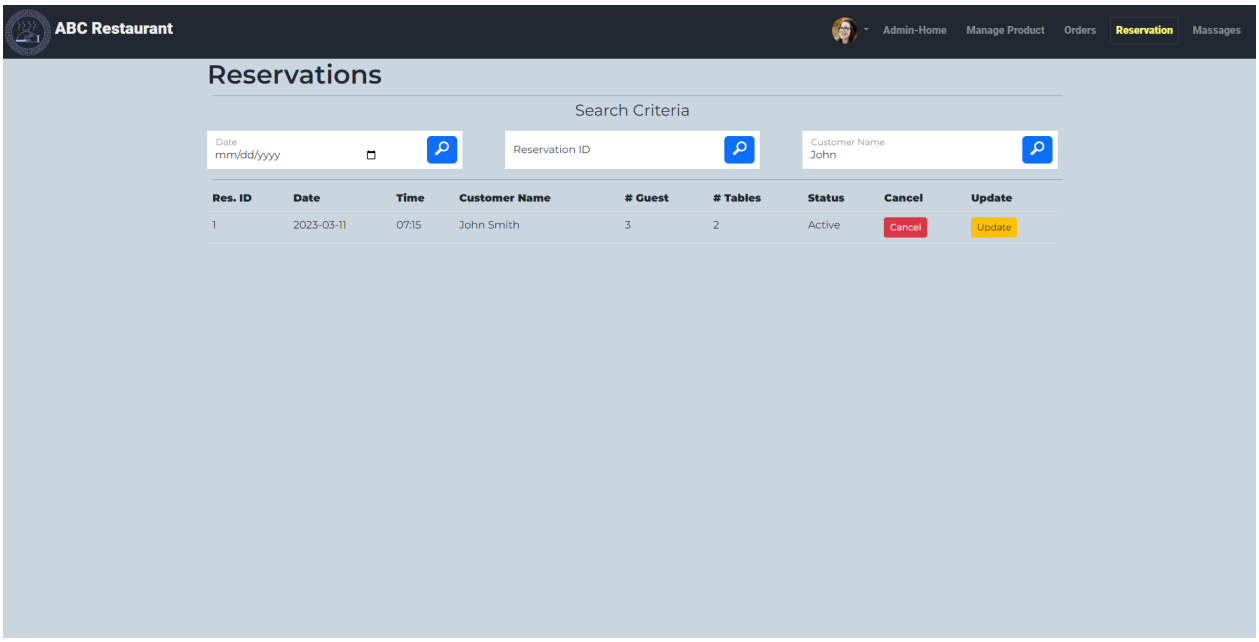
First Page - Employee Login



Description:

Using User ID; admin, Password: admin, an employee can log into the following admin portion of the restaurant where they can view/manage reservations.

Second Page - View/Manage Reservations

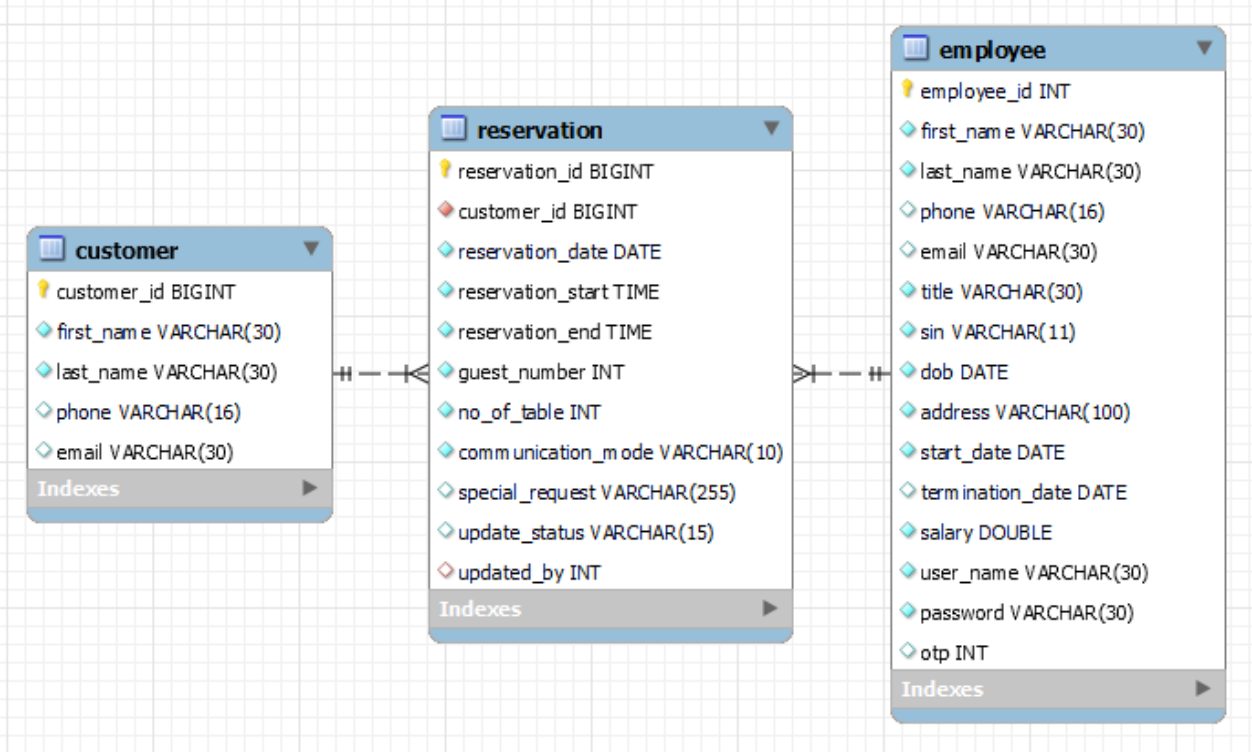


Description:

On this page employees can search for reservations by date, the reservation id or the customer's name. If a customer calls the restaurant and asks to update or cancel a reservation, an employee can use this page to implement the necessary changes which will also update the reservation in the database.

3. Database Entity Relationship Diagram & Schema Examples

ENTITY RELATIONSHIP DIAGRAM:



In the above diagram, the Reservation table has one foreign key, *customer_id*, which establishes the relationship of one to many between the customer and reservation table.

The employee table also has a one to many relationship with the customer table where one employee can update many reservations. The “updated_by INT” field in the reservation table would be filled by the corresponding *employee_id* to track which employee updated a reservation. In the reservation table the “updated_by INT” field is null by default, therefore we could not make *employee_id* a foreign key for the reservation table, as foreign keys must be “not null” by default. A possible solution to this would be to assign an employee to a reservation as soon as it is created, but at this stage of our project we decided it over complicated things.

Table: Customer

- (PK) customer_id

SQL Example:

```
-- create table customer
create table customer (
customer_id bigint auto_increment,
first_name varchar(30) not null,
last_name varchar(30) not null,
phone varchar(16) null,
email varchar (30) null,
constraint pk_customer primary key clustered (customer_id asc)
);
```

Table: Reservation

- (PK) reservation_id
- (FK) customer_id

SQL Example:

```
-- create table reservation
create table reservation (
reservation_id bigint auto_increment not null,
customer_id bigint not null,
reservation_date date not null,
reservation_start time not null,
reservation_end time not null, -- for breakfast duration 1.5 hours for dinner 2 hours
guest_number int not null,
no_of_table int not null,
communication_mode varchar(10) not null,
special_request varchar(255) null,
update_status varchar(15) null,
updated_by int null,
constraint pk_reservation primary key clustered (reservation_id asc),
constraint fk_reservation_customer foreign key (customer_id) references
customer(customer_id)
);
```

Table: Employee

- (PK) employee_id

SQL Example:

```
-- create table employee
create table employee (
employee_id int auto_increment,
first_name varchar(30) not null,
last_name varchar(30) not null,
phone varchar(16) null,
email varchar (30) null,
title varchar(30) not null,
sin varchar(11) not null,
dob date not null,
address varchar (100) not null,
start_date date not null,
termination_date date null,
salary double not null,
user_name varchar(30) not null,
password varchar(30) not null,
otp int null,
constraint pk_employee primary key clustered (employee_id asc)
);
```

Note:

Our project was developed with reference to the initial document provided (pictured below) and subsequently tailored to the instructions received on the afternoon of March 10th via Lea.

This project consists of a few parts, that needs to be delivered in a **zip file** by the end of Monday 6th June.

1- A Word(pdf) document that is explanation of the project. This document consists of these parts:

a. First paragraph gives a short description of the domain.

i. This domain is optional and can be anything like (daycare, carwash, library or anything you prefer)

b. Second paragraph is description of the web API that you are going to create and how it can be useful for this project and fulfill the requirements.

c. Third one is the list of routes and a short description of the reason of having each route.

2- A MOCKUP of the UI that is going to use this API with some explanations. This explanation can be on the mock-up picture or in a different document.

3- The Diagram and schema of the database and tables with the keys (PK, FK ,..) with their relationships. At least 3 tables needed for the project.