Abstract

To develop this project, skills learned from CPSC 471 were applied, including how to design an extended entity-relationship diagram (EERD), and converting the EERD into a relational model (RM). Before designing the EERD, a proposal was written, highlighting the importance of this project and how it can help small businesses such as massage clinics. From the EERD and RM, an object oriented diagram (OOD) was created to help with the implementation. During the implementation process, knowledge of SQL was applied to create schemas and tables, as well as modifying the tuples in the tables. Postman and stored procedures were used to test the functionality of the database. Making a front end website with a functional back end database helps the business spread word of their products and services, as well as giving the client and owner an easier time making business.

Introduction

During the tough times of COVID-19, smaller home run businesses such as massage clinics find it difficult to find customers to book appointments and conduct safe health screenings without a website. Friends and family that manages these small businesses keep track of appointments, clients, and health screenings in notebooks which is neither efficient nor organized. By developing a system with an easy to use website as the front end and a database in the backend, the process of booking appointments and health screenings will be much easier for both the client and admin. Clients will be able to use the front end for easy access to products and services offered, as well as finding a time to book the appointment. Administrators will have all the collected information from the client stored in the database for future reference.

System Description

This system API was coded in PHP hosted locally and implemented a MySQL database. The massage clinic file contains 4 files. The first is the config file. It is used to connect to the database. Next is the model file. It contains 1 file for every table in the relation model, each file is a class pertaining to a table and containing functions for the five operations view, search, insert, update and delete. Next is the API folder. It contains 15 folders, one folder is called authentication. This is used for logging in clients and employees. The other 14 each contain 5 files pertaining to the 5 different endpoints listed previously. Each of these files reads and writes json text.

Project Design

The two users for our system are the Employee and the Client. Some Employees have an admin login with all procedural privileges. Below is a complete transaction collection for all 3 user types.

Client

For the Client, the database stores account information as well as some personal information. When a Client is logged in they are authenticated and able to make some requests to the database. Once Authenticated they can view and search through products as well as view and search the calendar. They can also insert new health reports and search for health reports listed under their user id. The last thing they can do is book appointments by interesting them and they can search through appointments matching their user id.

Employee

For the employee, the database stores account information, some personal information as well as job & banking information. When an Employee logs in and is authenticated by the database, they have a few options. Firstly they can search through the client database. This does not sensitive personal information of the client. They can also search through bank accounts and find accounts tied to their user id. The last thing they can do is view and search Appointments. This appointment search is different from the client appointment search, instead of searching by client id, it searches by employee id and finds all appointments in the database that the employee is scheduled for.

Admin

The Admin is a special type of employee with their own special privileges. Once they login and are authenticated by the database they can insert, view, delete, and update virtually anything in the database with a few exceptions. The first exception is booking an appointment, an admin cannot book an appointment. Only a client can book an appointment, but an admin can still view, update or delete that appointment. The other exception is that an admin cannot insert a health report. Again only a client can create their own health report, while the admin can still view, update or delete that report.

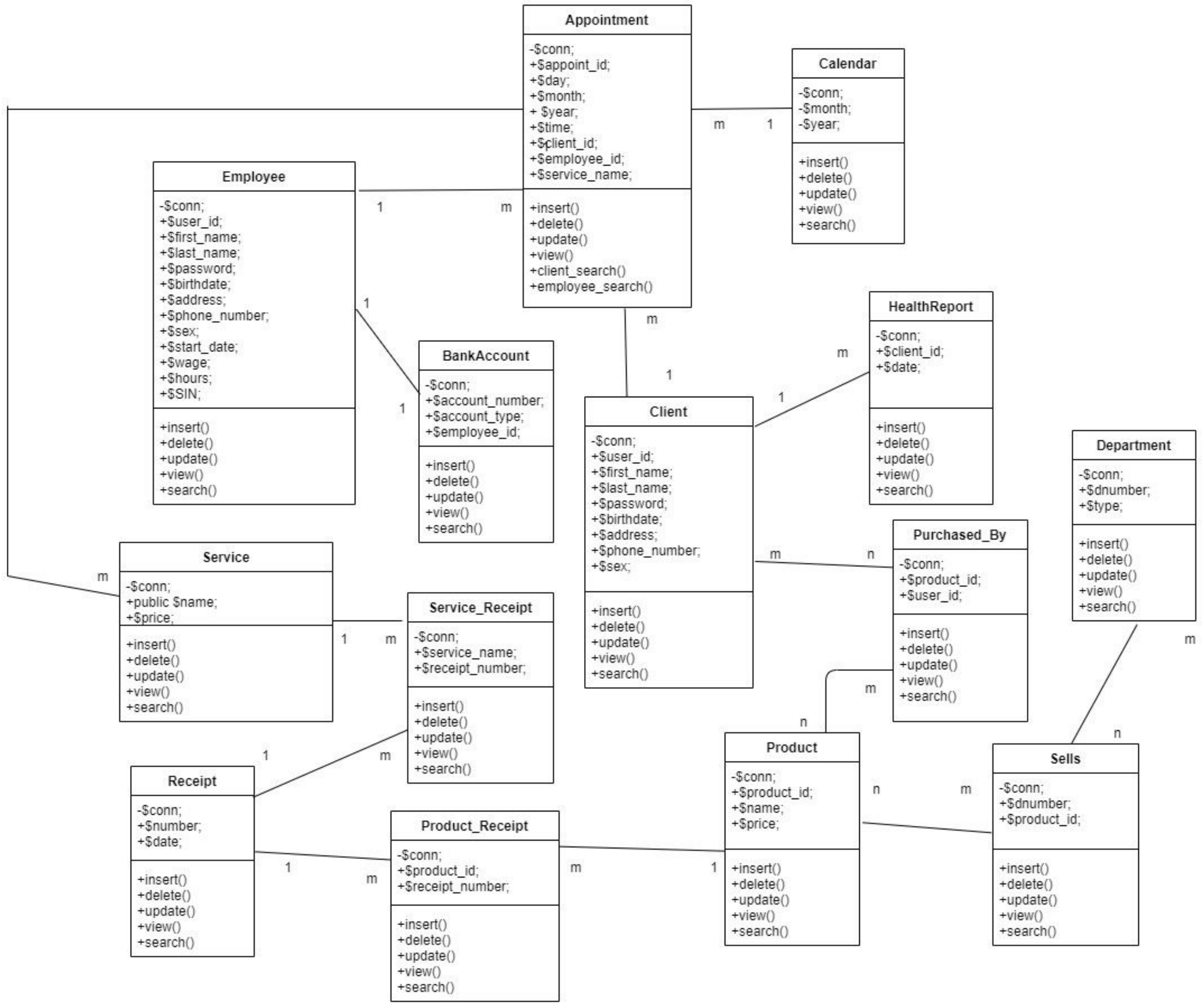
Extended Entity Relationship Diagram (Fig 1.0)

Implementation

Relational Model Diagram (Fig 1.1)

The relational model shown (fig 1.1) developed using PHP, hosted locally and implemented a MySQL database. The only change made when adapting the entity relation model to a relation model was the disjoint of the user entity. We decided to remove the user entity since client and user have the same attributes.

Object Oriented Model (Fig 1.2)



The object oriented model shown (fig 1.2) is modified from the original design for the system. In all the classes the function parameters have been removed because there are none in the final design for the system. Instead of passing the inputs as parameters, These classes are instantiated and then their variables are set before calling any functions. Then once the functions are called the desired stored procedure can be executed using the member variables.

SQL Queries
MySQL is used as the relational database management system.
The following queries are used to login users and assert that they are authorized to deploy all the other queries.

Authentication
Client Login: SELECT user_id FROM client WHERE user_id = id AND password = p;
Employee Login: SELECT user_id FROM employee WHERE user_id = id AND password = p;

The rest of the database contains stored procedures for each table. They follow
- View - a query to get all tuples of that table in the database.
- Search - a query to get only tuples matching the parameter(s) provided.
- Insert - a query to insert a new tuple into the database with provided parameter(s).
- Update - a query to update a tuple matching the parameter(s) provided using the provided parameter(s).
- Delete - a query to delete tuple(s) matching the parameter(s) provided.

Listed below are all the stored procedures contained in the database. Notice, some of the tables do not have update queries. These tables all have primary keys as their only attributes, so instead we update using two sets of data, first deleting one and inserting another.

Appointment
View: SELECT * FROM appointment;
Client Search: SELECT * FROM appointment WHERE client_id = ci;
Employee Search: SELECT * FROM appointment WHERE employee_id = ei;
Insert: INSERT INTO appointment SET day = d, month = m, year = y, time = t, client_id = ci, employee_id = ei, service_name = sn;
Update: UPDATE appointment SET day = d, month = m, year = y, time = t, client_id = ci, employee_id = ei, service_name = sn WHERE appoint_id = ai;
Delete: DELETE FROM appointment WHERE appoint_id = ai;

Bank Account
View: SELECT * FROM bank_account;
Search: SELECT * FROM bank_account WHERE employee_id = ei;
Insert: INSERT INTO bank_account SET account_number = an, account_type = at, employee_id = ei;
Update: UPDATE bank_account SET account_number = an, account_type = at WHERE employee_id = ei;
Delete: DELETE FROM bank_account WHERE employee_id = ei;

Calendar
View: SELECT * FROM calendar;
Search: SELECT * FROM calendar WHERE year = y;
Insert: INSERT INTO calendar SET month = m, year = y;
Delete: DELETE FROM calendar WHERE month = m AND year = y;

Client
View: SELECT user_id, first_name, last_name, birthdate, address, phone_number, sex FROM client;
Search: SELECT first_name, last_name, birthdate, address, phone_number, sex FROM client WHERE user_id = ui;
Insert: INSERT INTO client SET first_name = fn, last_name = ln, password = p, birthdate = bd, address = ad, phone_number = pn, sex = s;
Update: UPDATE client SET first_name = fn, last_name = ln, password = p, birthdate = bd, address = ad, phone_number = pn, sex = s WHERE user_id = ui;
Delete: DELETE FROM client WHERE user_id = ui;

Department
View: SELECT * FROM department;
Search: SELECT * FROM department WHERE dnumber = dn;
Insert: INSERT INTO department SET type = t;
Update: UPDATE department SET type = t WHERE dnumber = dn;
Delete: DELETE FROM department WHERE dnumber = dn;

Employee
View: SELECT * FROM employee;
Search: SELECT * FROM employee WHERE user_id = ui;
Insert: INSERT INTO employee SET first_name = fn, last_name = ln, password = p, birthdate = bd, address = ad, phone_number = pn, sex = s, start_date = sd, wage = w, hours = h, SIN = sin;
Update: UPDATE employee SET first_name = fn, last_name = ln, password = p, birthdate = bd, address = ad, phone_number = pn, sex = s, start_date = sd, wage = w, hours = h, SIN = sin WHERE user_id = ui;
Delete: DELETE FROM employee WHERE user_id = ui;

Health Report
View: SELECT * FROM health_report;
Search: SELECT * FROM health_report WHERE client_id = ci;
Insert: INSERT INTO health_report SET client_id = ci, date = d;;
Delete: DELETE FROM health_report WHERE client_id = ci AND date = d;

Product
View: SELECT * FROM product;
Search: SELECT * FROM product WHERE product_id = pi;
Insert: INSERT INTO product SET name = n, price = p;
Update: UPDATE product SET name = n, price = p WHERE product_id = pi;
Delete: DELETE FROM product WHERE product_id = pi;

Product Receipt
View: SELECT * FROM product_receipt;
Search: SELECT * FROM product_receipt WHERE receipt_number = rn;
Insert: INSERT INTO product_receipt SET product_id = pi, receipt_number = rn;
Delete: DELETE FROM product_receipt WHERE product_id = pi AND receipt_number = rn;

Purchased By
View: SELECT * FROM purchased_by;
Search: SELECT * FROM purchased_by WHERE user_id = ui;
Insert: INSERT INTO purchased_by SET product_id = pi, user_id = ui;
Delete: DELETE FROM purchased_by WHERE product_id = pi AND user_id = ui;

Receipt
View: SELECT * FROM receipt;
Search: SELECT * FROM receipt WHERE number = n;
Insert: INSERT INTO receipt SET date = d;
Update: UPDATE receipt SET date = d WHERE number = n;
Delete: DELETE FROM receipt WHERE number = n;

Sells
View: SELECT * FROM sells;
Search: SELECT * FROM sells WHERE dnumber = dn;
Insert: INSERT INTO sells SET dnumber = dn, product_id = pi;
Delete: DELETE FROM sells WHERE dnumber = dn AND product_id = pi;

Service
View: SELECT * FROM service;
Search: SELECT * FROM service WHERE name = n;
Insert: INSERT INTO service SET name = n, price = p;
Update: UPDATE service SET price = p WHERE name = n;
Delete: DELETE FROM service WHERE name = n;

Service Receipt

View: SELECT * FROM service_receipt;

Search: SELECT * FROM service_receipt WHERE receipt_number = rn;

Insert: INSERT INTO service_receipt SET service_name = sn, receipt_number = rn;

Delete: DELETE FROM service_receipt WHERE service_name = sn AND receipt_number = rn;

API Documentation

https://documenter.getpostman.com/view/13825322/TVmV7En2

User Manual

There are many ways to successfully run this project. Using XAMPP or AppServe should work. The requirement is that your system has Apache, MySQL and PHP.

Configuration

1) Configure XAMPP:
    a) Download the massageClinic folder and move to your desired directory
    b) Download XAMPP, open it and change Apache config httpd.conf lines 252 & 253 to the directory of where the project is (from previous step).
    Example:
    ```
    DocumentRoot "C:\Users\jaych\OneDrive\Desktop"
    <Directory "C:\Users\jaych\OneDrive\Desktop">
    ```
    c) Start Apache module and MySQL module.

2) To create the database on your local machine:
    a) Create a schema called 'massage_clinic'
    b) Run massage_clinic.sql in an SQL query
    c) Go to massageClinic/config/Database.php and change lines 6-7 to your local database credentials

3) To run the website on your machine:
    a) Open any browser (Chrome works best)
    b) Type 'http://localhost/massageClinic/frontend/nonuser/home.php' and enter

4) To test the endpoints on Postman on the database:
    a) Import 'Massage_Clinic_API.postman_collection.json' into Postman
    b) Most of the commands are preloaded with inputs already for testing, just press 'send' to run a command.
    c) The changes are reflected in the database, refresh the corresponding table to see the results