Rate my Doctor: Final Report

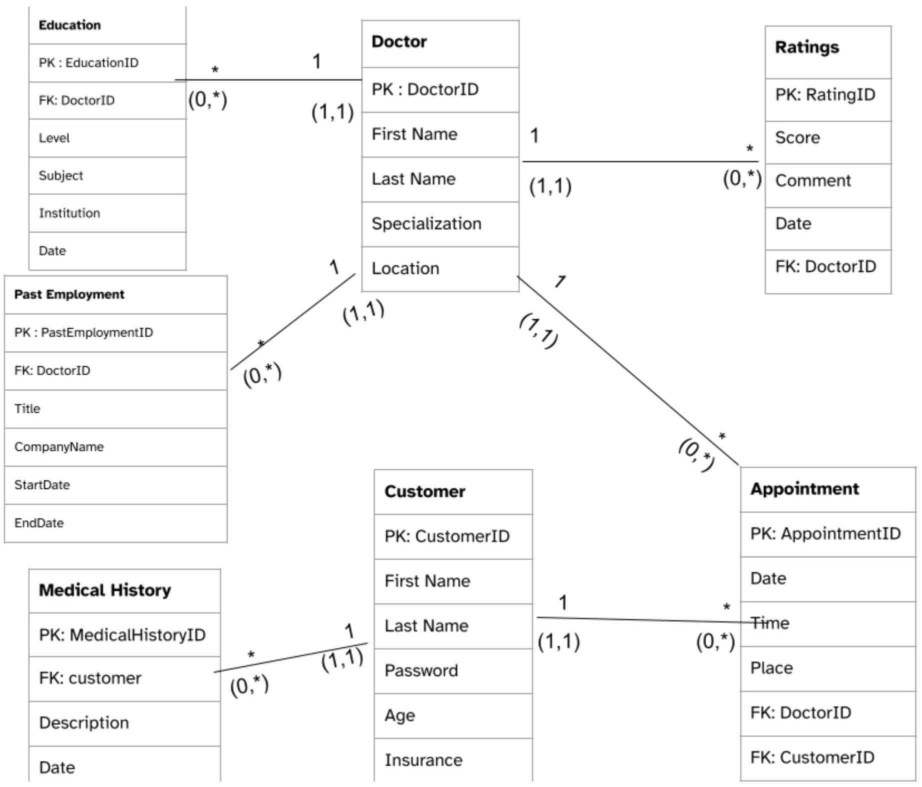
By: Rohan Govathoti, Viet Nguyen, Vincent Wen

Link to GitHub

https://github.com/vivelaviet/RateMyDoctor

Changes to Project Design

Final Project Database Design ERD



Doctor (DoctorID, First Name, Last Name, Specialization) Partial Dependencies(DoctorID- > First Name, Last Name, Specialization)
Education(EducationID, DoctorID, Level, Subject, ,Institution, Date) Partial Dependencies(EducationID -> DoctorID,Level, Subject, ,Institution, Date)
Past Employment(PastEmploymentID, DoctorID, CompanyName, Title, DateStart, DateEnd) Partial Dependencies(PastEmployementID -> DoctorID, CompanyName)
Ratings (RatingID, Score, Comment, Date, DoctorID) Partial Dependencies(RatingID -> DoctorID, Comment)
Customer(CustomerID, Password, Age, Insurance, First Name, Last Name) Partial Dependencies(CustomerID - > Age, Insurance, First Name, Last Name)
Medical History(MedicalHistoryID, CustomerID, Description, Date) Partial Dependencies(MedicalHistoryID -> CustomerID, Date)
Appointment(AppointmentID, Date, Time, Place, DoctorID, CustomerID) Partial Dependencies(AppointmentID -> DoctorID, CustomerID, Time, Date)

This ERD was what we submitted in our Final Project Database Design and it was implemented into our current design, meaning all the tables shown above have been created in our database. However, even though all of the tables shown above were created in the database, not all of them were used. We decided to adjust our approach and not use the Education, Past Employment, and Medical History tables. As a group, we came to the conclusion that these tables provided extra information which was outside the scope of the project. The main tables that we worked with and integrated into our web application were Doctor, Customer, Ratings, and Appointment.

Project Files

- *.gitignore*
  - Viet Nguyen
  - Made just to ignore files in the xampp htdocs directory.
- db.php
  - Viet Nguyen
  - Simple php code to be included on most of the pages so that the connection to SQL isn't redundant.

- *Login.php*
  - Viet Nguyen
  - Implemented the three roles so that there is a variable that can be used within the session. This means that the roles can persist between pages. This was supposed to be for our fourth member but they are not showing up so we made do with simply buttons that picked your user role instead of a login/register page.

- *View_Appointment.php*
  - Vincent Wen
  - The view appointment queries through the database using SELECT SQL command and echo the appointment table onto the page. This page is shown in all the user however in customer view point only the admin/scheduler can delete the row in the appointment table using SQL command DELETE.

- *addcustomers.php*
  - Vincent Wen
  - This allows customers to add/update their information into the web application for the user customer/admin using SQL command INSERT and UPDATE of rows in the customer table.

- *adddoctors.php*
  - Viet Nguyen
  - This handles INSERT and UPDATE of rows in the doctor table.

- *addratings.php*
  - Rohan Govathoti created the Add/Update Ratings tab in the web application which allows for ratings to be added and/or updated on the web application and database.This handles INSERT and UPDATE of rows in the ratings table.

- *appointment.php*
  - Vincent Wen
  - This allows the scheduler to add/update their information into the web application for the user scheduler/admin using SQL command INSERT and UPDATE of rows in the customer table.

- *contact.php*
    - Vincent Wen
    - More for contact information but this was a extra view page that doesn't do anything on the backend side

- *header.php*
    - Viet Nguyen
        - Header that displays all the tabs. it also uses session variables to hide certain tab from different users.
    - Vincent Wen
        - Help with the header for different users in displaying the different tabs for each user.

- *index.php*
    - Rohan Govathoti implemented the search functionality on the index page. This allows for a doctor to be searched by their first name, last name, location, or a combination of the attributes.

- *ratemydoctor.sql*
    - Rohan Govathoti created the database structure: includes all the tables, keys, and relationships that are used in the web application.
    - SQL file to import to create the sample table
    - Viet Nguyen included views and indexes

- *viewcustomer.php*
    - Vincent Wen added a view of the customer information which comes from the attributes in the customer table from the SQL connection.
    - This part handles the SELECT and DELETE of rows in the customer table.

- *viewdoctors.php*
    - Viet Nguyen added a simple view doctor functionality that displays the attributes of the doctor table from the SQL connection.
    - This handles SELECT and DELETE of rows in the doctor table.

- *viewratings.php*
    - Rohan Govathoti created the View All Ratings tab which queries the database to display all rating attributes. This handles SELECT and DELETE of rows in the ratings table.

Project Functionalities

- Each project must have at least three user types.
  - The first user type will be an admin.
  - Your group can select the last two user types.

Viet… (Originally for the fourth member but they left and didn't work on anything so the implementation is bare buttons and just three buttons that represent the user roles). This was using session variables which persist when you switch off tabs using session_start() function of php. Then on every other page that uses different role views, they can check these session variables to see what the current role is, and echo the appropriate html if they are allowed to see it.

- Functionality Set 1: Each project must allow users to have an account on their website. The account must have the following functionality:
  - A registration method that is appropriate to the user type.
  - The ability to login into the user account and access proper functionality.
  - The ability to update relevant profiles.
  - The ability to delete the account and all related appropriate data.

Vincent

- Functionality Set 2: Each project must have a scheduling functionality. The appropriate users must be able to:
  - Create appointments
  - View appointments
  - Update appointments
  - Delete appointments

Vincent worked on the Functionality Set 2. To implement the appointment we needed to first finish the customer and doctor table as those primary are the foreign for the appointment table. In appointment.php allows for scheduler/admin to create and update appointments. To insert a row into the appointment table, the scheduler/admin would insert the DATE,TIME,doctor, and customer. However, for the doctor and customer I implemented a drop down menu that query through the doctor or customer table and grab all the first and last name, and display the option to pick for the scheduler/admin using it. For update, there is a drop down menu made for update to choose which date and place to update then reenter the information. The SQL commands used were INSERT and UPDATE to insert a row into the database. In the View_Appointment.php, gives a view of the appointments for all users to see however only the scheduler/admin have access to delete the appointment. Using the SQL command select to view and the SQL command DELETE to delete a row.

Viet

- Functionality Set 3: Each project must have an item or experience functionality that allows the users to:
    - Add items or experiences
    - View items or experiences
    - Update items or experiences
    - Delete items or experiences

For this feature, this is the doctors for the most part. Customers, Admins, and Schedulers can see the doctor tab. You can add doctors, and edit their information using update and select in the add doctors tab, but this is only made available to the admin since it doesn't make sense for schedulers or users to add it at this point. There are drop downs for doctor IDs that are labeled with their First Name and Last Name so that it is easy to tell what the user is updating rather than staring at doctor IDs.

Vincent works on the implementation of customer. Addcustomer.php can be found in the customer and admin. In the php, only customer/admin can add customers using attributes First Name, Last Name, Age, Insurance, and password into the customer table. Using the drop down menu to update previous information added. In the viewcustomer.php, all the users are able to see this menu however only the customer and admin can delete the row in the customer table.

Rohan

- Functionality Set 4: Each project must have a comment, status, or review functionality that allows the users to:
    - Add comments, statuses, or reviews
    - View comments, statuses, or reviews
    - Update comments, statuses, or reviews
    - Delete comments, statuses, or reviews

Rohan implemented Functionality Set 4. First, Rohan created viewratings.php which queries the Ratings table to display all attributes of all rating entities. The "RatingID", "Score", "Comment", "Date", and "Doctor Name" are displayed on the viewratings.php page. Additionally on the viewratings.php page, there is a button which deletes the rating entity when clicked. On viewratings.php, users are able to view reviews as well as delete reviews. Rohan also created addratings.php which provides text input fields and input field drop downs which allow users to enter information for a ratings entity. The user has the option to add a rating entity by inputting information into the "Score", "Comment", "Date", selecting a Doctor in the "Doctor Name" dropdown, and clicking the Add button which will create an entry for the rating entity on the viewratings.php page.

The user also has to option to update an existing rating entity by selecting an entry in the “For Update Use” drop down, providing input for all the following fields, and clicking the Update button which will update the results on the viewratings.php page. On addratings.php, users are able to add new ratings as well as update any existing ratings.

Extra Addendum

- Kun, our fourth member was supposed to handle one of the functionalities, so we made it simple by simply using buttons to do roles
- Views were sort of unnecessary in our projects for the most part as we didn’t use many joins.

Indexes:

```
ALTER TABLE `appointment`
  ADD PRIMARY KEY (`AppointmentID`),
  ADD UNIQUE KEY `FastAppointment` (`Place`),
  ADD KEY `DoctorID` (`DoctorID`),
  ADD KEY `CustomerID` (`CustomerID`);

--
-- Indexes for table `customer`
--
ALTER TABLE `customer`
  ADD PRIMARY KEY (`CustomerID`),
  ADD UNIQUE KEY `FastCustomer` (`FirstName`,`LastName`);

--
-- Indexes for table `doctor`
--
ALTER TABLE `doctor`
  ADD PRIMARY KEY (`DoctorID`),
  ADD UNIQUE KEY `FastDoctorSearch` (`FirstName`,`LastName`,`Location`,`Specialization`);

--
-- Indexes for table `education`
--
ALTER TABLE `education`
  ADD PRIMARY KEY (`EducationID`),
  ADD KEY `DoctorID` (`DoctorID`);
```

```
--
-- Indexes for table `medicalhistory`
--
ALTER TABLE `medicalhistory`
  ADD PRIMARY KEY (`MedicalHistoryID`),
  ADD KEY `CustomerID` (`CustomerID`);

--
-- Indexes for table `pastemployment`
--
ALTER TABLE `pastemployment`
  ADD PRIMARY KEY (`PastEmploymentID`),
  ADD KEY `DoctorID` (`DoctorID`);

--
-- Indexes for table `ratings`
--
ALTER TABLE `ratings`
  ADD PRIMARY KEY (`RatingID`),
  ADD UNIQUE KEY `FastRatings` (`Comment`,`RatingID`) USING BTREE,
  ADD KEY `DoctorID` (`DoctorID`);
```