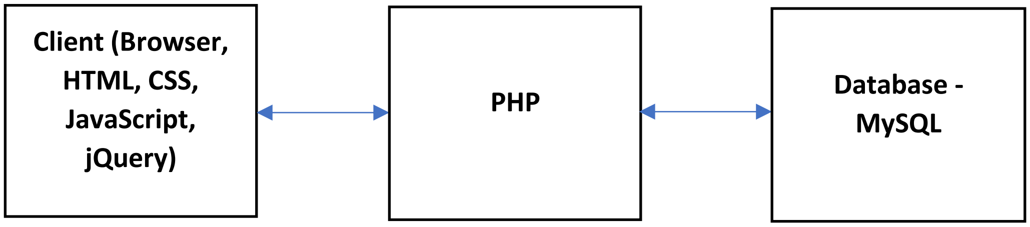
Introduction:

Contact Management System involves the creation of a database host application that interfaces with a backend SQL database implementing a Contact List application.

Architecture:

Application uses a three-tier-architecture. A client (browser), a web server and the database.



Client is built using HTML, CSS, jQuery and JavaScript. Web-Server is built using PHP scripting and the database used is MySQL.

Client is used to request and insert data into in the database. The request is sent to the web-server to process the request and make the appropriate query to the database. The database sends the data as a response to the fetch query and success message for an update or insert request.

Design and Assumptions:

Initial Setup:

- Contacts.CSV file is separated according to different tables as per given schema.
- The CSV file is then imported into the MYSQL database into 4 tables – Contact, Address, Phone and Date.

Searching Contacts: (API: /?search=query)

- The searching is accomplished using a single search box which searches through the contact names, address fields, phone numbers and event dates.
- On the back-end, there is query joining all 4 tables to contact table on contact ID.
- The contact ID, First Name, Middle Name and Last Name are displayed as search results.
- Modify/Delete button are provided for each contact which appears as search result.

Adding new Contact: (API: /add)

- The fields displayed are First Name, Middle Name and Last Name, Address fields – Type, address line, city, state and zip, Phone fields – Phone Type and Phone number, Event fields – Event Type and Event Date.
- Only Names fields are required:
- The form data is POSTed in JSON format to the server. On the back-end, the form data is received and parsed using the $_POST method in PHP. DB queried are made through mysqli() command in PHP.
- On the DB, first the contact names are inserted into the contact table, then using the contact ID (Foreign Key), the address table phone table and date tables are filled with the data entered by the user.

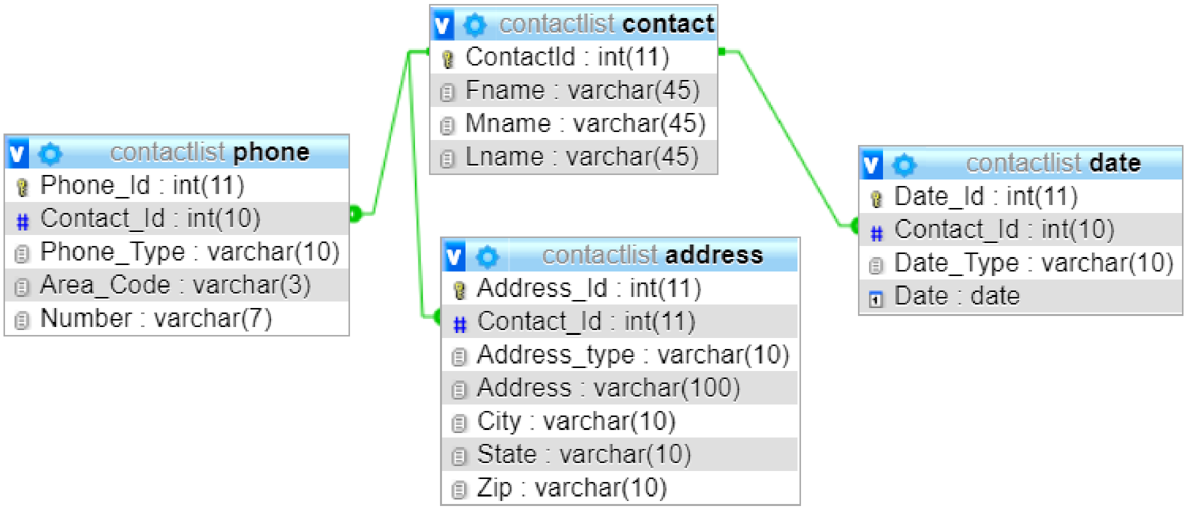
Modify Contact: (API: /modify/:id)

- Modify button are provided for each contact which appears as search result.
- The fields are same as Adding a contact, additionally the user will be able to delete existing address fields, phone fields and event fields or add new fields.
- When the user modifies an address, phone or date field, the modified data is send along with the table unique ID as an HTML hidden input element. On the back-end, the corresponding table with unique ID is modified.
- When the user deletes an address, phone or date field, a ajax call asynchronously deletes the record in DB table according to the given ID.

Delete Contact: (API: /delete/:id)

- Delete button are provided for each contact which appears as search result.
- On the back-end, the address, phone and event information for the contact is deleted first, then the contact. This prevent foreign key violations. This is done using ajax call.

Entity – Relationship Diagram:



contactlist **contact**
ContactId : int(11)
Fname : varchar(45)
Mname : varchar(45)
Lname : varchar(45)

contactlist **phone**
Phone_Id : int(11)
Contact_Id : int(10)
Phone_Type : varchar(10)
Area_Code : varchar(3)
Number : varchar(7)

contactlist **address**
Address_Id : int(11)
Contact_Id : int(11)
Address_type : varchar(10)
Address : varchar(100)
City : varchar(10)
State : varchar(10)
Zip : varchar(10)

contactlist **date**
Date_Id : int(11)
Contact_Id : int(10)
Date_Type : varchar(10)
Date : date