----------------------------------------------------------------

Creating a Database: Database Name: **assigndb**

----------------------------------------------------------------

We will create 3 tables:

**users**:

The users table is designed to store detailed information about each user, including their name, email, location, and geographical coordinates. Here's the explanation of each column:

* **id int(11) NOT NULL)**:
  + A unique identifier for each user.
  + The column type int(11) indicates it can store integers up to 11 digits.
  + It is the primary key for this table
  + This column is required (NOT NULL).
* **name (varchar(100) NOT NULL)**:
  + Stores the name of the user.
  + The varchar(100) data type allows a maximum of 100 characters to be stored.
  + The NOT NULL constraint ensures that a user’s name must be provided when the record is created.
* **email (varchar(255) NOT NULL)**:
  + Stores the email address of the user.
  + The varchar(255) data type allows for emails of up to 255 characters.
  + The NOT NULL constraint ensures that an email address is mandatory when inserting a new user record.
* **location (varchar(125) DEFAULT NULL)**:
  + Stores the physical location of the user (e.g., city or country).
  + The varchar(125) data type allows up to 125 characters.
  + The DEFAULT NULL clause means that this field is optional and can be left empty.
* **latitude (decimal(10,8) NOT NULL)**:
  + Stores the geographic latitude of the user.
  + The decimal(10,8) data type ensures that the value can have up to 10 digits in total, with 8 digits after the decimal point (precision for geographic coordinates).
  + The NOT NULL constraint means that latitude must be provided for each user.
* **longitude (decimal(11,8) NOT NULL)**:
  + Stores the geographic longitude of the user.
  + The decimal(11,8) data type ensures that the value can have up to 11 digits in total, with 8 digits after the decimal point.
  + The NOT NULL constraint means that longitude must be provided for each user.
* **created\_at (timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP)**:
  + Stores the timestamp of when the user record is created.
  + The DEFAULT CURRENT\_TIMESTAMP ensures that this field is automatically set to the current date and time when a new record is inserted.
* **updated\_at (timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP)**:
  + Stores the timestamp of when the user record is last updated.
  + The DEFAULT CURRENT\_TIMESTAMP ensures this field is initially set to the current date and time when the record is created.
  + The ON UPDATE CURRENT\_TIMESTAMP clause ensures that this field is automatically updated to the current timestamp whenever the record is modified.

**user\_sites**:

user\_sites table is designed to manage the relationship between users and sites. A user can be assigned to multiple sites, and this table helps store those associations.

**id (int(11) NOT NULL)**:

A unique identifier for each record in the user\_sites table.

The int(11) data type indicates that it can store integers up to 11 digits.

This column will typically serve as the primary key.

**user\_id (int(11) NOT NULL)**:

This column stores the ID of the user associated with the site.

The int(11) data type matches the id type in the users table.

The NOT NULL constraint ensures that a user ID must be provided when creating a record in this table.

This column will typically be a foreign key referencing the id column in the users table, establishing the relationship between the user\_sites and users tables.

**site\_id (int(11) NOT NULL)**:

This column stores the ID of the site that is assigned to the user.

The int(11) data type matches the id type in the sites table.

The NOT NULL constraint ensures that a site ID must be provided when creating a record in this table.

This column will typically be a foreign key referencing the id column in the sites table, linking users to specific sites.

**created\_at (timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP)**:

* This field records the timestamp of when the user-site association is created.
* The DEFAULT CURRENT\_TIMESTAMP ensures that this value is automatically set to the current date and time when a new record is inserted.

**sites**:

This table will store all site information.

**id int(11) NOT NULL)**:

* A unique identifier for each user.
* It is of type int, which can store integer values.
* This column will likely be set as the primary key to ensure each user is uniquely identifiable.

**site\_name (varchar(100) NOT NULL):**

* This field stores the name of the site the user is associated with.
* The varchar(100) data type means it can hold up to 100 characters.
* The NOT NULL constraint indicates that this field is mandatory when creating a user record.

**site\_link (varchar(2083) DEFAULT NULL)**:

* This field stores the URL or link of the site associated with the user.
* The varchar(2083) data type allows for long URLs (up to 2083 characters, which is the maximum URL length in some browsers).
* This field is optional, as indicated by the DEFAULT NULL clause, meaning it can be left empty when inserting or updating a record.

**created\_at (timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP)**:

* This field records the timestamp of when the user record is created.
* The DEFAULT CURRENT\_TIMESTAMP ensures that this value is automatically set to the current date and time when a new record is inserted.

**updated\_at (timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP)**:

* This field tracks the timestamp of the last update made to the user record.
* The DEFAULT CURRENT\_TIMESTAMP ensures that this value is initially set to the current date and time when the record is created.
* The ON UPDATE CURRENT\_TIMESTAMP clause ensures that this field is automatically updated to the current timestamp whenever any changes are made to the record.

------------------------------------------------------------------------------------------

**Web Pages Information & Files are located folder is /src**

-------------------------------------------------------------------------------------------

**Creating a MySQLi Database Connection Class:**

I can use a MySQLi database class , Once the connection is established, this file will be included on all pages.

**User List Display:**

We will create a User.php class. In this class, we will instantiate the database connection class. Once the connection is established, we can prepare a query to fetch the list of users along with the sites they are assigned to.

**Filter Functionality:**

We will add filters to display users who are geographically close to the current user. The user can pass location data (latitude, longitude, and radius) as query string parameters to filter the results.

Example: http://localhost/vaave/src/user.php?latitude=17.43258940&longitude=78.40706910&radius=5

**User Details and Assigned Sites:**

The user list page will have a clickable username that links to a page displaying the sites the user is assigned to. Since a user can be linked to multiple sites, clicking on the username will show the information about the assigned sites.

**Opening Links in a New Tab:**

When the user clicks on a site link, it will open in a new target tab.

**API Information (These files will be stored in the \api folder)**

**Creating UpdateUser.php:**

This API file will allow updating user information such as name, email, and location. The database connection class will be used to establish the connection, and the pass the parameters will be used Mysqli DB class function update function. The response format will be in JSON format for API testing (using Postman).

**Creating Search Functionality (Geolocation-based):**

An API endpoint will be created to search for users who are geographically close to the current user, based on latitude, longitude, and a specified radius. The database connection class will be used to fetch the user list within the given radius.

**Get Nearby Locations Function:**

We will create a getNearbyLocations() function to filter users based on the specified radius and geolocation (latitude, longitude). This function will calculate the distance between users and return a list of users within the specified radius.

**Checking the API Endpoint**

* 1. **UpdateUser.php**
  2. **searchByLocation.php**

For testing the API endpoints, the Postman collection folder will be shared.

File will be available on Documentation Folder