**NEARBY ALUMNI SEARCH**

Nearby Alumni Search is implemented using PHP Backend APIs.

The objective of the project is to create two API endpoints to locate the nearby alumni present in the radius of 10km. The PHP-MySQLi-Database-Class ORM is used to handle the database interactions.

The two endpoints are:

* Endpoint 1: To update a user’s details, such as name, email, and location.
* Endpoint 2: To retrieve a list of alumni who are geographically close to the current user, based on a specified radius.

Endpoint 1: To update a user’s details, such as name, email, and location.

Purpose:

* If the user is new user, then register the user must register.
* If the user is already register, then if he wants to update any information about him then he can update or he can directly get the alumni’s present within the radius of 10km.
* The location of the user updates every time he visits the application.
* The PHP-MySQLi-Database-Class ORM is used to handle the database interactions like insertion, deletions, updating etc.

Endpoint 2: To retrieve a list of alumni who are geographically close to the current user, based on a specified radius.

* The JSON application format is used to send the response.
* This endpoint is used to retrieve the nearby alumni’s present with in the radius of 10kms.
* The alumni’s retrieved are present within at least one alumni network with the user.
* Haversine formula is used to determine the distance between the user and alumni who are in common alumni.

Database Design:

* The database in this project contains three tables.
  + Users: contains the details of user (name, email, location).
  + Networks: contains all alumni networks present.
  + User\_networks: contains the user id’s and network id’s and this describes the many-many relationships between user and networks.
* When the user registers, the details are stored in the Users table.
* After the user registers, then all the alumni’s he is present are stored in the Networks table which does not contains any duplicates.
* The User\_networks table stores the many-many relationships between the users and networks i.e., user id and network id.

**Major Assumptions**

* Email is unique identifier.
* Location provided by GPS via browser.
* Each user can belong to multiple networks.

**Query Structure:**

To retrieve the nearby alumni’s, I have used the rawsql query which is supported by the ORM.

Query:

SELECT u.id, u.name, u.email, u.latitude, u.longitude

    FROM users u

    JOIN user\_networks un ON u.id = un.user\_id

    WHERE un.network\_id IN (" . implode(',', $networkIds) . ")

    AND u.id != ?

    GROUP BY u.id

    HAVING (

        6371 \* acos(

            cos(radians(?)) \* cos(radians(u.latitude)) \*

            cos(radians(u.longitude) - radians(?)) +

            sin(radians(?)) \* sin(radians(u.latitude))

        )

    ) <= ?

I used this query to calculate the distance between the alumni’s and user using haversine formula.

The query performs a **JOIN** with user\_networks to include only users from the same alumni networks.

GROUP BY u.id is used to eliminate duplicates if a user is in multiple common networks.

The query uses **prepared statements** (? placeholders) to avoid SQL injection.