Affilomania Dev Exercise

The purpose of this exercise is to test the approach to coding.

Tips for the exercise:

1. Your code is clean and readable.
2. You **do not** use comments in your code, but instead fallow a clear naming conventions.
3. Your code is follows the OOP best practices.
4. You fulfilled ALL the requirements.
5. You can use (if you like) any PHP or / and Angular framework, scaffolding or other code generator you like as long as you add init instructions so we can run your code.

Exercise:

Please create a simple CRUD application that Creates, Reads, Updates and Deletes a User from a MySql database

Coding Standard Requirements:

- All the communication between front end and backend should be done by AJAX calls.
- All GET (Read) Actions should return a JSON.
- All POST/PUT/DELETE action must send a json type body (type header should be "application/json") and return response in JSON format (even if just status code and message).
- All Requests should between front-end and back-end should be non blocking (async)
- Please Use php language for the backend service (preferably with a framework of your choosing like: Laravel, Lument, Symfony, etc but not mandatory)

Requirements:

- A table should display all data of all existing users ordered by "last_update_time" (**no pagination is required**) – see below "User Model".

- Clicking on a user row will open a form populated with the selected user current details where user details can be edited with tow buttons: **Save and Delete**.
  a. Click on Save will update the user details in the database and refresh the table.
  b. Click on Delete will delete the user from the database and refresh the table.

- Above the table should be an "Add" button that opens the same form as the "edit" action, but without any populated details.

  the form will have tow buttons: Save and Cancel
  a. Click on Save creates a new user in the database and refresh the table

b. Click on Cancel clear the form and close it.

- All Required validations (see below: “Data Requirements”) should be in place and checked both on front-end (if possible) and backend. An error should be presented to user if one of the fields failed validation. (**You can use simple alert for validation errors**)

User Model:

User table fields:

- **Id** - auto incremented integer
- **first_name** - string(45)
- **last_name** – string(45)
- **email** – string(254) unique
- **username** – string(45) unique
- **password** – md5 hashed password
- **created_at** - UNIX timestamp
- **last_update_time** – dateTime, default NOW
- **is_active** – Boolean, default True

Data Requirements

- **email** – must be an email pattern (example a@a.a)
- **password** – must have at least 1 lowercase alphabetical char, 1 uppercase alphabetical char, 1 numeric char, 1 special char (like ^&$%...) and must be at least 8 char long.
- **All fields from the user model are mandatory**

Submit Exercise Instructions:

- Please pack all the result code in a single .zip file and send to daniel@affilomania.com with the title **Dev Test – {your name}**
- The Zip file must include:
  a. all the code.
  b. sql dump file (with the database structure).
  c. short README.md file that explain how to deploy your solution so we can test run it.

*** If you choose you can send git repository instead of .zip file (NOT MANDATORY)**

**** For any questions please contact me on daniel@affilomania.com**