**CWEB 280 Final Project API Documentation**

**Names:** Dominic Broley and Kalan Kalynka

**API Documentation:**

Our plan on using Postman, we will use basic URL’s such as GET, POST, PUT, and DELETE to achieve the necessary requirements, and functionality to fulfill the application. Below are some of the following implementations we plan on using:

The main local host route that will be used is for the server: http://127.0.0.1:8080/

The main local host that will be used for the Vue.js is the: http://localhost:5173/

**Vue.js:**  
**Component – UserOnPage (Fetches from backend)**

* **GET Request**
  + **Route:** 127.0.0.1:8080/userInDB/userOnPage
  + **Method:** GET
  + **Description:** Fetches all users from the database.
  + **Parameters:** None.
  + **Response:**
    - Success (200): JSON array of user objects.
    - Error (500): Send a JSON message from the server and says “Server Error.”
* **PUT Request**
  + **Route:** 127.0.0.1:8080/userInDB/updateUser/:id
  + **Method:** PUT
  + **Description:** Updates the details of an existing user, which is their name and access level.
  + **Parameters:** 
    - id: User ID to update, through path parameter.
    - Username (string): The name the user can change it to.
    - accessLevel (string): The selected access level for the user. (e.g., read, write, admin).
  + **Response:**
    - Success (200): Send a JSON message from the server and says, “User updated successfully.”
    - Error (404): Send a JSON message from the server and says, “User not found.”
    - Error (500): Send a JSON message from the server and says “Server Error.”
* **DELETE Request**
  + **Route:** 127.0.0.1:8080/userInDB/updateUser/:id
  + **Method:** DELETE
  + **Description:** Deletes a user by their ID.
  + **Parameters:**
    - id: User ID to delete, through path parameter.
  + **Response:** 
    - Success (200): Send a JSON message from the server and says, “User deleted successfully.”
    - Error (404): Send a JSON message from the server and says, “User not found.”
    - Error (500): Send a JSON message from the server and says “Server Error.”

**Backend:**

**Register Page:**

* **GET Request:** 
  + **Route:** 127.0.0.1:8080/register
  + **Method:** GET
  + **Description:** Loads the registration form for user account creation.
  + **Parameters:** None.
  + **Response:** Loads pug form page for user registration.
* **POST Request:** 
  + **Route:** 127.0.0.1:8080/register
  + **Method:** POST
  + **Description:** Submits registration details to create a new user profile.
  + **Parameters**:
    - username (string): User’s chosen username.
    - password (string): User’s password.
    - accessLevel (string): The selected access level for the user. (e.g., read, write, admin)
  + **Response:** JSON message indicating success or failure of the registration.

**Login Page:**

* **GET Request:** 
  + **Route:** 127.0.0.1:8080/login
  + **Method:** GET
  + **Description:** Loads the login page, for the user to fill in their username and password (if they have an account/profile).
  + **Parameters:** None.
  + **Response:** Success if the information that they provided is valid.
* **POST Request:** 
  + **Route:** 127.0.0.1:8080/login
  + **Method:** POST
  + **Description:** Verifies login credentials against the user database and assigns the appropriate access level.
  + **Parameters:**
    - username (string): User’s username.
    - password (string): User’s password.
* **Response:** JSON message indicating login success or failure. If successful, the response includes the user's **AccessLevel** (e.g., read, write, admin) to define their permissions.

**User Profile Management**

* **PUT Request**:
  + **Route:** 127.0.0.1:8080/user/update-username
  + **Method:** PUT
  + **Description**: Updates the user's username.
  + **Parameters**:
    - newUsername (string): The new username.
    - newPassword (string): The new password.
* **Response**: JSON message indicating the update status.
* **DELETE Request**:
  + **Route:** 127.0.0.1:8080/user/delete-username
  + **Method:** DELETE
  + **Description:** Deletes the user’s profile/account.
  + **Parameters:** None.
  + **Response:** JSON message indicating deletion status.

**Comment Page**

* **GET Request:**
  + **Route**: 127.0.0.1:8080/weather/comments
  + **Method**: GET
  + **Description**: Fetches all comments to display on the page.
  + **Parameters**: None.
  + **Response**: A list of all comments in JSON format.
* **GET Request:**
  + **Route:** 127.0.0.1:8080/weather/comments/edit/:id
  + **Method**: GET
  + **Description**: Gets the edit form for the comment.
  + **Authentication Required**: Yes (verifyToken, hasWriteAccess).
  + **Parameters**: None.
  + **Response**: Success message, redirect to comments page.
* **POST Request:**
  + **Route**: 127.0.0.1:8080/weather/comments
  + **Method**: POST
  + **Description**: Posts a new comment to the page.
  + **Authentication Required**: Yes (verifyToken, hasWriteAccess).
  + **Parameters**:
    - commentText (string): Text of the comment.
  + **Response**: Success message, redirect to comments page.
* **PUT Request:**
  + **Route**: 127.0.0.1:8080/weather/comments/edit/:id
  + **Method**: PUT
  + **Description**: Updates an existing comment with the provided ID. From the GET request of /weather/edit/:id form. It will then update the specific reply.
  + **Authentication Required**: Yes (verifyToken, hasWriteAccess).
  + **Parameters**:
    - id (string): The ID of the comment to update.
    - commentText (string): The new comment text.
  + **Response**: Success message, redirect to comments page.
* **DELETE Request:**
  + **Route**: 127.0.0.1:8080/weather/comments/delete/:id
  + **Method**: DELETE
  + **Description**: Deletes a comment with the provided ID.
  + **Authentication Required**: Yes (verifyToken, hasWriteAccess).
  + **Parameters**:
    - id (string): The ID of the comment to delete.
  + **Response**: Success message, redirect to comments page.

**Replies within the Comment Page**

* **GET Request:**
  + **Route**: 127.0.0.1:8080/weather/comments/reply/:id
  + **Method**: GET
  + **Description**: Fetches all replies for a specific comment.
  + **Parameters**: None.
  + **Response**: A list of all replies in JSON format.
* **GET Request:**
  + **Route:** 127.0.0.1:8080/weather/comments/reply/edit/:id
  + **Method:** GET
  + **Description:** Display edit form for reply.
  + **Authentication Required**: Yes (verifyToken, hasWriteAccess).
  + **Parameters:** None.
  + **Response:** Loads the reply page and if they are authenticated then success.
* **POST Request:**
  + **Route**: 127.0.0.1:8080/weather/comments/reply/:id
  + **Method**: POST
  + **Description**: Adds a reply to a specific comment.
  + **Parameters**:
    - commentId (string): The ID of the comment to reply to.
    - replyText (string): The text of the reply.
* **Response**: Success message, redirect to comments page.
* **PUT Request:**
  + **Route**: 127.0.0.1:8080/weather/comments/reply/:id
  + **Method**: PUT
  + **Description**: Updates an existing reply with the provided ID. From the GET request of /weather/reply/:id/edit form. It will then update the specific reply.
  + **Authentication Required**: Yes (verifyToken, hasWriteAccess).
  + **Parameters**:
    - id (string): The ID of the reply to update.
    - replyText (string): The new reply text.
* **Response**: Success message, redirect to comments page.