API Documentation: Verify User API

This documentation provides a detailed explanation of the `verify_user` API for verifying user credentials in a database. The API is implemented in PHP and uses a MySQL database to validate user credentials.

Overview

The `verify_user` API accepts `user_id` and `user_password` as input and verifies whether the provided credentials match a user in the database.

Base URL

The API is hosted at your server endpoint (e.g., `http://localhost/verify_user.php`).

HTTP Methods

POST

- **Purpose**: Verifies if a user exists in the database and checks the validity of the provided password.

Input Parameters

The API expects a JSON payload with the following fields:

Field	Type	Description	Required
`user_id`	`string`	The unique identifier of the user.	Yes
`user_password`	`string`	The password associated with the `user_id`.	Yes

Sample Request

```
json
Copy code
{
    "user_id": "3333",
    "user_password": "tanimvai"
}
```

Responses

The API returns a JSON response based on the verification result.

Success Response

If the credentials are valid, the API returns:

```
json
Copy code
{
    "user_id": "3333",
    "user_name": "Tanim Vai"
}
```

Error Responses

1. **Invalid Credentials**:

   ```
   json
   Copy code
   {
       "error": true,
       "message": "Invalid user_id or password"
   }
   ```

2. **Missing Parameters**:

   ```
   json
   Copy code
   {
       "error": true,
       "message": "Missing user_id or user_password"
   }
   ```

3. **Unsupported HTTP Method**:

   ```
   json
   Copy code
   {
       "error": true,
       "message": "Method not allowed"
   }
   ```

Code Explanation

Database Connection

```
php
Copy code
$host = "localhost";
$username = "root";
$password = "";
$db_name = "attendance_system";

$conn = new mysqli($host, $username, $password, $db_name);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

- Establishes a connection to the MySQL database.
- Terminates with an error if the connection fails.

Input Handling

```
php
Copy code
$data = json_decode(file_get_contents("php://input"), true);
```

- Reads and decodes the JSON payload sent in the POST request.

Credential Verification

1. **SQL Query**:

   ```
   php
   Copy code
   $sql = "SELECT user_id, user_name, user_password FROM users WHERE
   user_id = '$user_Id'";
   $result = $conn->query($sql);
   ```

   - Fetches the `user_id`, `user_name`, and `user_password` from the database where `user_id` matches the input.
2. **Check Query Result**:

   ```
   php
   Copy code
   if ($result && $result->num_rows > 0) {
       $row = $result->fetch_assoc();
   }
   ```

    - Ensures a user exists with the given `user_id`.

3. **Password Validation**:

   ```
   php
   Copy code
   if ($user_password === $row['user_password']) {
       echo json_encode([
           "user_id" => $row['user_id'],
           "user_name" => $row['user_name']
       ]);
   }
   ```

    - Compares the provided password with the password stored in the database. **(For security, replace this with `password_verify` if passwords are hashed)**.

4. **Error Handling**:
    - If the user is not found or the password does not match, appropriate error messages are returned.

Default Case

```
php
Copy code
default:
    http_response_code(405);
    echo json_encode(["error" => "Method not allowed"]);
    break;
```

- Handles unsupported HTTP methods by returning a `405 Method Not Allowed` status.

Database Schema

Table Name: `users`

Column Name	Data Type	Description
`user_id`	`VARCHAR`	Unique identifier for users.
`user_name`	`VARCHAR`	Name of the user.
`user_email`	`VARCHAR`	Email of the user.
`user_password`	`VARCHAR`	User password (plain or hashed).

Security Considerations

1. **Use Prepared Statements**:
    - Replace `"$user_Id"` in SQL queries with prepared statements to prevent SQL injection.

    ```
    php
    Copy code
    $stmt = $conn->prepare("SELECT user_id, user_name, user_password FROM
    users WHERE user_id = ?");
    $stmt->bind_param("s", $user_Id);
    ```

2. **Hash Passwords**:
    - Store passwords using hashing (`password_hash`) for better security.

    ```
    php
    Copy code
    $hashedPassword = password_hash($user_password, PASSWORD_BCRYPT);
    ```

3. **Validate Input**:
    - Ensure all required fields are present and sanitized.

Usage Example

Request

```
bash
Copy code
curl -X POST http://localhost/verify_user.php \
-H "Content-Type: application/json" \
-d '{
    "user_id": "3333",
    "user_password": "tanimvai"
}'
```

Response

```
json
Copy code
{
    "user_id": "3333",
    "user_name": "Tanim Vai"
}
```