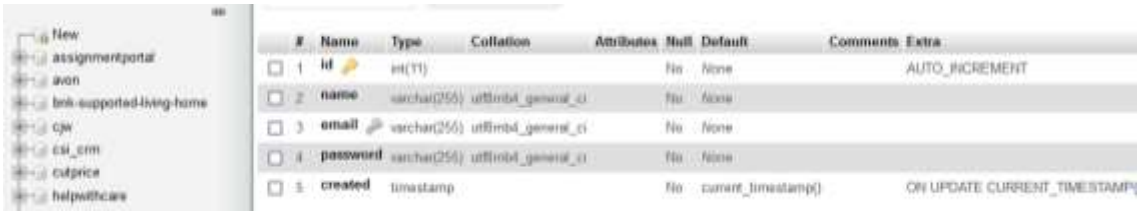
PHP REST API CRUD with JWT Authentication

Step 1: Create a Database

- Either create the database manually or from the phpmyadmin with any name like **phprestapi**

Step 2 Create a Table

- Either create the table manually or from the phpmyadmin with any name like **register** in **phprestapi** database



Step 3 Create a directory for configuration

- Create the folder in XAMP/P -> htdocs with any name like **PHP REST API CRUD with JWT Authentication**
- Open this folder and create api folder and necessary files if required like index, style, script etc.
- Open api folder and create config folder.
- Open config folder and create a new file called database.php

```php
<?php
class Database
{

    //Specify Database Credential
    private $host = "localhost";
    private $dbName = "phprestapi";
    private $userName = "root";
    private $password = "";

    public $conn;

    //get the Database Connection
    public function getConnection()
    {
        $this->conn = null;

        try {
            $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" .
```

```
                $this->dbName, $this->userName, $this->password);
            $this->conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

        } catch (PDOException $exception) {
            echo "Connection error: " . $exception->getMessage();
        }
        return $this->conn;
    }
}
```

Step 4 Create user

- Create the objects folder inside api folder and create a new file called user.php

```
<?php
class User
{
    private $conn;
    private $table = "register";
    // object properties
    public $id;
    public $name;
    public $email;
    public $password;

    public function __construct($db)
    {
        $this->conn = $db;
    }

    public function create()
    {
        $this->name = htmlspecialchars(strip_tags($this->name));
        $this->email = htmlspecialchars(strip_tags($this->email));
        $this->password = htmlspecialchars(strip_tags($this->password));

        $this->password = password_hash($this->password, PASSWORD_DEFAULT);

        $query = "insert into " . $this->table . " (name,email,password)
values(?,?,?)";

        try {
            $stmt = $this->conn->prepare($query);
            $run = $stmt->execute([$this->name, $this->email, $this-
>password]);

            return true;
        } catch (PDOException $exception) {
```

```
            if ($exception->getCode() === '23000') {
                echo json_encode(array("Duplicate Entry" => $this->email . "
already exists"));die;
            } else {

                echo json_encode(array("insertDataSQLError" => $exception-
>getMessage()));die;
            }
        }
    }
}
```

Step 5 Create API for user

- Create user folder and Create a new file called create_user.php and Following steps to be performed for this

    1. We need to set headers on this new file.
    2. Connect to database and register table
    3. Assign submitted data to object properties
    4. Use the create () method

```
<?php
// required headers
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers,Authorization, X-Requested-With");

// files needed to connect to database
include '../config/database.php';
include '../objects/user.php';

// get database connection
$database = new Database();
$db = $database->getConnection();

// instantiate user object
$user = new User($db);

// get posted data
$json = json_encode($_POST);
$data = json_decode($json);

// set user property values
```

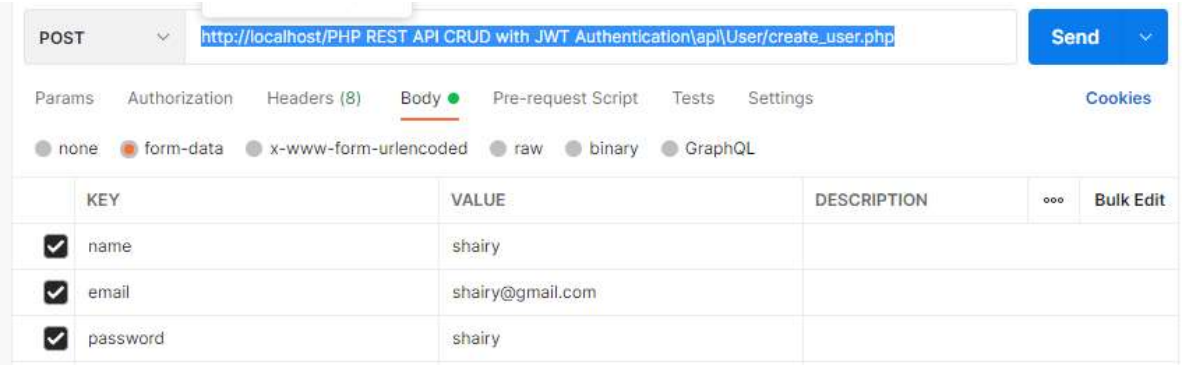
```php
$user->name = $data->name;
$user->email = $data->email;
$user->password = $data->password;

// create the user
if (
    !empty($user->name) &&
    !empty($user->email) &&
    !empty($user->password) &&
    $user->create()
) {

    // set response code
    http_response_code(200);
    // display message: user was created
    echo json_encode(array("message" => "User was created."));
}

// message if unable to create user
else {

    // set response code
    http_response_code(400);
    // display message: unable to create user
    echo json_encode(array("message" => "Unable to create user."));
}
```

Step 6 Output for insert user

To test for the successful creation of a data, open POSTMAN. Enter the following as the request URL

http://localhost/PHP REST API CRUD with JWT Authentication\api\User \create_user.php

- Click the "form-data" tab.
- Enter key and value
- Send



Step 7 Read Data

- After successfully insertion of user, now we **read the complete data or single user data**. For this we will add 2 functions in user.php of object folder.

```
// Read Data
    public function read()
    {
        //query to read all data
        $query = "select * from " . $this->table;
        try {
            // Prepare query
            $stmt = $this->conn->prepare($query);
            // Execute query
            $stmt->execute();
            return $stmt;
        } catch (PDOException $exception) {
            echo json_encode(array("Student Read Data Error" => $exception-
>getMessage()));die;
        }
    }
    //Read Single Student Data
    public function readSingleUser($id)
    {
        //query to read single student data
        $query = "select * from " . $this->table . " where id=" . $id;
        try {
            // Prepare query
            $stmt = $this->conn->prepare($query);

            // Execute query
            $stmt->execute();
            // set the resulting array to associative
            $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
```

```
            if ($stmt->rowCount() > 0) {
                return $stmt;
            } else {
                return false;
            }
        } catch (PDOException $exception) {
            echo json_encode(array("Single Student Read Data Error" =>
$exception->getMessage()));die;
        }
    }
```

Step 8 Create API for read single and all data

- Create a new file called read_user.php in user folder and Following steps to be performed for this
    1. We need to set headers on this new file.
    2. Connect to database and register table
    3. Assign submitted data to object properties
    4. Use the read () and readSingleUser() method

```
<?php
// required headers
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers,Authorization, X-Requested-With");

// files needed to connect to database
include '../config/database.php';
include '../objects/user.php';

// get database connection
$database = new Database();
$db = $database->getConnection();

// instantiate user object
$user = new User($db);

// get posted data
$json = json_encode($_POST);
$data = json_decode($json);

if (isset($_GET["id"])) {
    $stmt = $user->readSingleUser($_GET["id"]);
```

```
    if ($stmt) {
        $row = $stmt->fetchAll();
        http_response_code(200);
        echo json_encode(array("message" => "Login Successfully.", "data" =>
$row));
    } else {
        http_response_code(401);
        echo json_encode(array("message" => "Invalid Credential."));
    }
} else {
    $stmt = $user->read();
    $rowcount = $stmt->rowCount();
    if ($rowcount > 0) {
        // set response code - 200 found
        $data = array();
        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
            extract($row);
            array_push($data, array(
                "id" => $row["id"],
                "name" => $row["name"],
                "email" => $row["email"],
                "password" => $row["password"],
            ));
        }
        http_response_code(200);
        echo json_encode(array("message" => "Read data Successfully.", "data"
=> $data));

    } else {
        http_response_code(401);
        echo json_encode(array("message" => "issue in Read data Request."));
    }
}
```

Output for insert user

To read all and single user data, open POSTMAN. Enter the following as the request URL

http://localhost/PHP REST API CRUD with JWT Authentication\api\ User\ read_user.php

http://localhost/PHP REST API CRUD with JWT Authentication\api \User\ read_user.php? Id=3

Step 9 Update Data

- After view/read user, now we **update the user data**. For this we will add update functions in user.php of object folder

```
    //Update User Data
    public function update($id)
    {
        $query = "update " . $this->table . " set name=?,email=?,password=?
where id= " . $id;
        try {
            $stmt = $this->conn->prepare($query);
            $run = $stmt->execute([$this->name, $this->email, $this-
>password]);
            return true;
        } catch (PDOException $e) {
            {
                echo json_encode(array("UpdateDataSQLError" => $e-
>getMessage()));die;
            }
        }
    }
```

Step 10 Create API for update data

- Create a new file called update_user.php in user folder and Following steps to be performed for this
    1. We need to set headers on this new file.
    2. Connect to database and register table
    3. Assign submitted data to object properties
    4. Use the update method

```
<?php
// required headers
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers, Authorization, X-Requested-With");

// files needed to connect to database
include '../config/database.php';
include '../objects/user.php';

// get database connection
$database = new Database();
```

```php
$db = $database->getConnection();

// instantiate user object
$user = new User($db);

// get posted data
$json = json_encode($_POST);
$data = json_decode($json);

echo json_encode(array("data" => $data, "id" => $_GET["id"]));
// make sure data is not empty
if (
    !empty($data->name) &&
    !empty($data->email) &&
    !empty($data->password)
) {

    // set data property values
    $user->name = $data->name;
    $user->email = $data->email;
    $user->password = $data->password;

    // update the data
    if ($user->update($_GET["id"])) {

        // set response code - 201 created
        http_response_code(201);
        // tell the user
        echo json_encode(array("message" => "user data is updated."));
    }
    // if unable to create the data, tell the user
    else {
        // set response code - 503 service unavailable
        http_response_code(503);
        // tell the user
        echo json_encode(array("message" => "Unable to update data."));
    }
}

// tell the user data is incomplete
else {
    // set response code - 400 bad request
    http_response_code(400);
    // tell the user
    echo json_encode(array("message" => "Unable to update data. Data is
incomplete."));
}
```

Step 11 Delete Data

- Last, but not the least we will **delete the user data**. For this we will add delete functions in user.php of object folder.

```
// Delete User
    public function delete($id)
    {
        $query = "delete from " . $this->table . " where id= " . $id;
        try {
            $stmt = $this->conn->prepare($query);
            $stmt->execute();
            return true;
        } catch (PDOException $e) {
            {
                echo json_encode(array("DeleteUserSQLError" => $e-
>getMessage()));die;
            }
        }
    }
```

Step 12 Create API for delete data

- Create a new file called delete_user.php in user folder and Following steps to be performed for this
    1. We need to set headers on this new file.
    2. Connect to database and register table
    3. Assign submitted data to object properties
    4. Use the delete method

```
<?php
// required headers
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers, Authorization, X-Requested-With");

// files needed to connect to database
include '../config/database.php';
include '../objects/user.php';
```

```php
// get database connection
$database = new Database();
$db = $database->getConnection();

// instantiate user object
$user = new User($db);

// get posted data
$json = json_encode($_POST);
$data = json_decode($json);

// delete user from the db
if (isset($_GET['id'])) {
    $stmt = $user->delete($_GET["id"]);

    // set response code - 201 created
    http_response_code(201);

    // tell the user
    echo json_encode(array("message" => "user is deleted."));
}

// if unable to delete the quiz, tell the user
else {

    // set response code - 503 service unavailable
    http_response_code(503);

    // tell the user
    echo json_encode(array("message" => "Unable to delete user."));
}
```

Step 13 Register User with JWT

A. Create a new file called register.php in user folder

B. Download composer using command in api folder: **composer require firebase/php-jwt**

C. create new file core. php in config folder

D. Copy all the code of step5- Create API for user and use below lines in the starting of register. php files

<?php

require '..\vendor\autoload.php';

use Firebase\JWT\JWT;

include_once '../config/core.php';

?>

E. and the following insert code change and use JWT

```
// create the user

if (
    !empty($user->name) &&
    !empty($user->email) &&
    !empty($user->password) &&
    $user->create()
) {
    // set response code - 201 created
    http_response_code(201);

    array_push($payload, array('data' => array("email" => $user->email)));
    $jwt = JWT::encode($payload, $key);
    // tell the user

    echo json_encode(array("message" => "Registration successfully.", "token"
=> $jwt));
}

// message if unable to create user
else {

    // set response code
    http_response_code(400);
    // display message: unable to create user
    echo json_encode(array("message" => "Unable to Register user."));
}
```

F. The complete code is below

```
<?php

require '..\vendor\autoload.php';
use Firebase\JWT\JWT;
include_once '../config/core.php';

// required headers
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers,Authorization, X-Requested-With");

// files needed to connect to database
include '../config/database.php';
```

```php
include '../objects/user.php';

// get database connection
$database = new Database();
$db = $database->getConnection();

// instantiate user object
$user = new User($db);

// get posted data
$json = json_encode($_POST);
$data = json_decode($json);

// set user property values
$user->name = $data->name;
$user->email = $data->email;
$user->password = $data->password;

// create the user
if (
    !empty($user->name) &&
    !empty($user->email) &&
    !empty($user->password) &&
    $user->create()
) {
    // set response code - 201 created
    http_response_code(201);

    array_push($payload, array('data' => array("email" => $user->email)));
    $jwt = JWT::encode($payload, $key);
    // tell the user

    echo json_encode(array("message" => "Registration successfully.", "token"
=> $jwt));
}

// message if unable to create user
else {

    // set response code
    http_response_code(400);
    // display message: unable to create user
    echo json_encode(array("message" => "Unable to Register user."));
}
```

Step 14 Login User with JWT

A. create new file core. php in config folder

B. Create login function in user.php

```
//Login Student
    public function readLogin($email, $password)
    {
        $query = "Select * from " . $this->table . " where email= '" . $email
. " ' LIMIT 1";
        try {
            $stmt = $this->conn->prepare($query);
            $stmt->execute();
            return $stmt;
        } catch (PDOException $e) {
            echo json_encode(array("AuthenticationStudentSQLError" => $e-
>getMessage()));die;
        }

    }
```

C. Copy all the code of step8- Read API for user

D. The complete code is below

```
<?php
// required headers
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers, Authorization, X-Requested-With");

require '../vendor/autoload.php';
use Firebase\JWT\JWT;
include_once '../config/core.php';
include "../config/database.php";
include "../objects/user.php";

$database = new Database();
$db = $database->getConnection();

$user = new User($db);

// get posted data
$json = json_encode($_POST);
$data = json_decode($json);
```

```php
//Read data with JWT
if (
    !empty($data->email) &&
    !empty($data->password)
) {
    $stmt = $user->readLogin($data->email, $data->password);
    $rowcount = $stmt->rowCount();

    if ($rowcount > 0) {
        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
            extract($row);
            if (password_verify($data->password, $password)) {

                array_push($payload, array('data' => array("email" => $user->email)));
                $jwt = JWT::encode($payload, $key);
                http_response_code(200);
                echo json_encode(array("message" => "Login Successfully.", "token" => $jwt));
            } else {
                http_response_code(401);
                echo json_encode(array("message" => "Username/Password Incorrect."));

            }
        }
    } else {
        http_response_code(401);
        echo json_encode(array("message" => "Username/Password Incorrect."));

    }
} else {
    http_response_code(400);
    echo json_encode(array("message" => "Unable to Find Student. Data is incomplete."));
}
```

Step 15 AJAX Call from the Frontend to this API

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div class="container">
  <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-
bs-target="#register">
    Register
  </button>
  <button type="button" class="btn btn-primary" data-bs-toggle="modal" data-
bs-target="#login">
    Login
  </button>
</div>

<!-- The Register Modal -->
<div class="modal" id="register">
  <div class="modal-dialog">
    <div class="modal-content">
      <!-- Modal Header -->
      <div class="modal-header">
        <h4 class="modal-title">Registration Form</h4>
        <button type="button" class="btn-close" data-bs-
dismiss="modal"></button>
      </div>

      <!-- Modal body -->
        <div class="modal-body">
            <form id="signup" method="post">
                <div class="col-md-12">
                    <input class="form-control mt-3" type="text" name="name"
placeholder="Name" required >
                </div>
                <div class="col-md-12">
                    <input class="form-control mt-3" type="email" name="email"
placeholder="E-mail Address" required>
                </div>
                <div class="col-md-12">
                    <input class="form-control mt-3" type="password"
name="password" placeholder="Password" required>
```

```
                </div>
                <div class="col-md-12">
                    <input class="form-control mt-3" type="tel"
name="number"  placeholder="Contact Number" required>
                </div>
                <div class="col-md-12">
                    <input class="form-control mt-3" type="text"
name="companyName" placeholder="Your Company Name" required >
                </div>
                <div class="col-md-12">
                    <select name="country" class="form-select mt-3" required>
                            <option value="0">Country</option>
                            <option value="USA">USA</option>
                            <option value="CANADA">CANADA</option>
                            <option value="UK">UK</option>
                            <option value="RUSSIA">RUSSIA</option>
                            <option value="INDIA">INDIA</option>
                    </select>
                </div>
                <button type="submit" id="regbutton" class="btn btn-primary
mt-3">Submit</button>
            </form>
        </div>
    </div>
  </div>
</div>


<!-- The Login Modal -->
<div class="modal" id="login">
  <div class="modal-dialog">
    <div class="modal-content">
      <!-- Modal Header -->
      <div class="modal-header">
        <h4 class="modal-title">Login Form</h4>
        <button type="button" class="btn-close" data-bs-
dismiss="modal"></button>
      </div>

      <!-- Modal body -->
        <div class="modal-body">
            <form id="loginn" method="post">
                <div class="col-md-12">
                    <input class="form-control mt-3" type="email" name="email"
placeholder="E-mail Address" required>
                </div>
                <div class="col-md-12">
                    <input class="form-control mt-3" type="password"
name="password" placeholder="Password" required>
```

```
                </div>
                <button type="submit" id="loginbutton" class="btn btn-primary
mt-3">Submit</button>
            </form>
        </div>
    </div>
  </div>
</div>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min
.js"></script>
<script>
$(document).ready(function(){
  $("#regbutton").click(function(e){
        e.preventDefault();
        let regForm = document.getElementById("signup");
        let regFormData = new FormData(regForm);
        $.ajax({
          url:"http://localhost/Digitizemydesign/api/Client/register.php",
          type: "POST",
          data: regFormData,
          contentType: false,
          processData: false,
          success: function (res) {
            console.log(res);
            alert(res.message);
            window.location.reload();
          },
        error: function (xhr, status, data) {
              let err = eval("(" + xhr.responseText + ")");
              alert(err.message);
          }
        });
  });
});
</script>
<script>
$(document).ready(function(){
  $("#loginbutton").click(function(e){
        e.preventDefault();
        let loginForm = document.getElementById("loginn");
        let loginFormData = new FormData(loginForm);
        $.ajax({
          url:"http://localhost/Digitizemydesign/api/Client/login.php",
          type: "POST",
          data: loginFormData,
          contentType: false,
```

```
          processData: false,
          success: function (res) {
            console.log(res);
            alert(res.message);
           // window.location.reload();
          },
        error: function (xhr, status, data) {
              let err = eval("(" + xhr.responseText + ")");
              alert(err.message);
          }
        });


  });
});
</script>
</body>
</html>
```