UNIVERSITI TEKNOLOGI MALAYSIA

SCHOOL OF COMPUTING

SESSION 2020/2021 SEMESTER 2

SCSJ3483-01 Web Technology

Group Sparta, Final Documentation

Name:

Muhammad Teruyuki Bin Ikuo @ Mohamad Alif Ikuo (A18CS0164)

Ahmad Nuri Bin Mohd Khalili (A18CS0115)

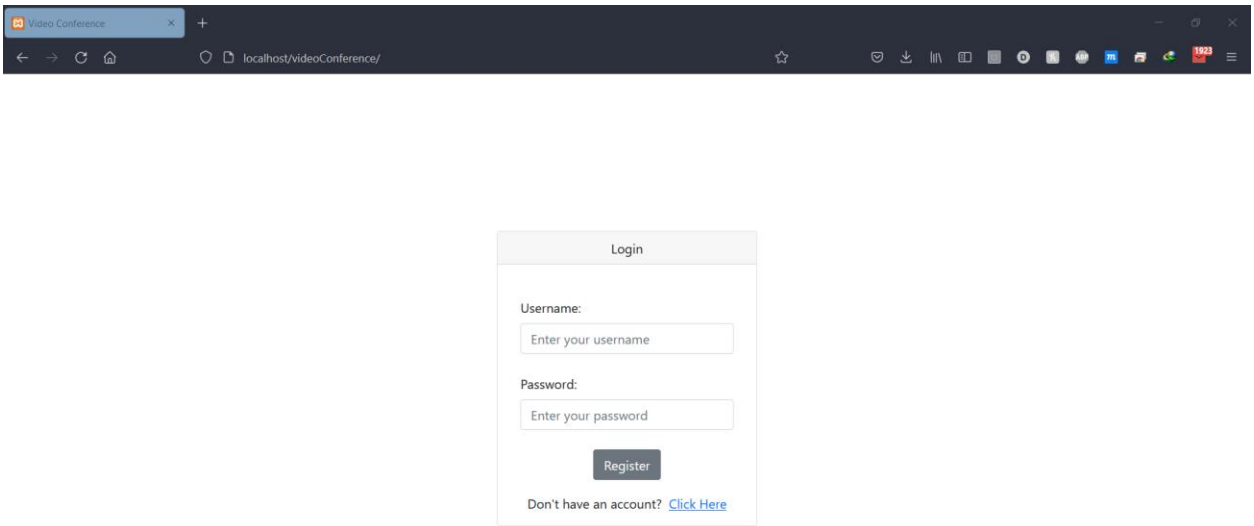
Ahmad Syahir Bin Abdul Hanim (A18CS0017)

Dzil Hafizin Bin Mazlan (A18CS0054)

Video Conference

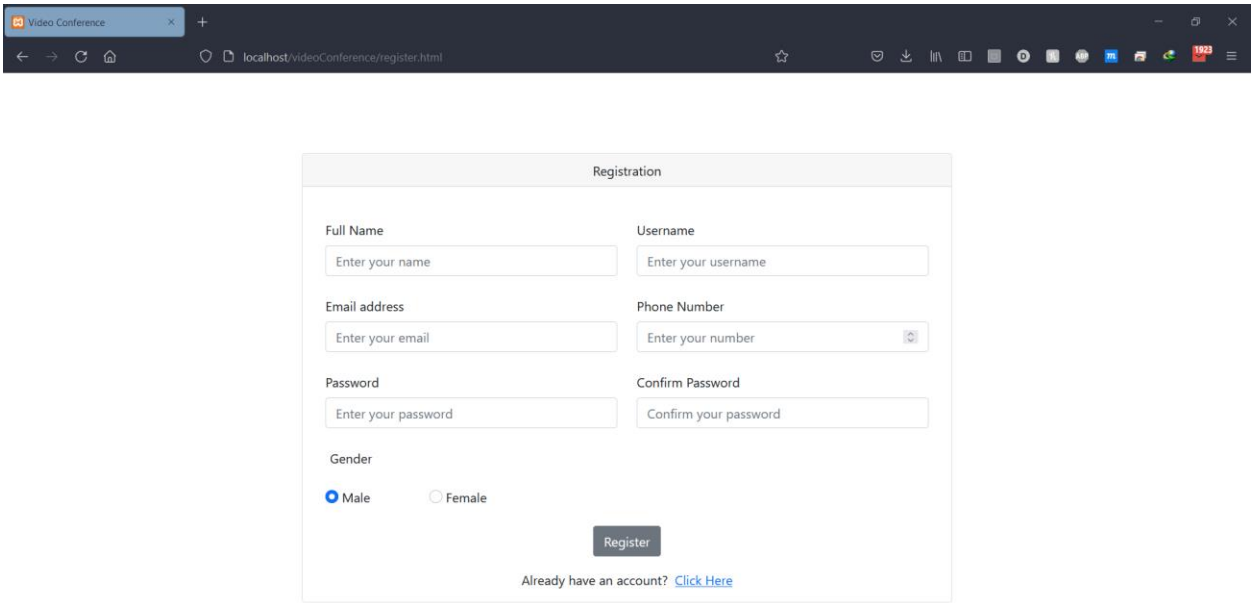
User Interface:

1. Login Page



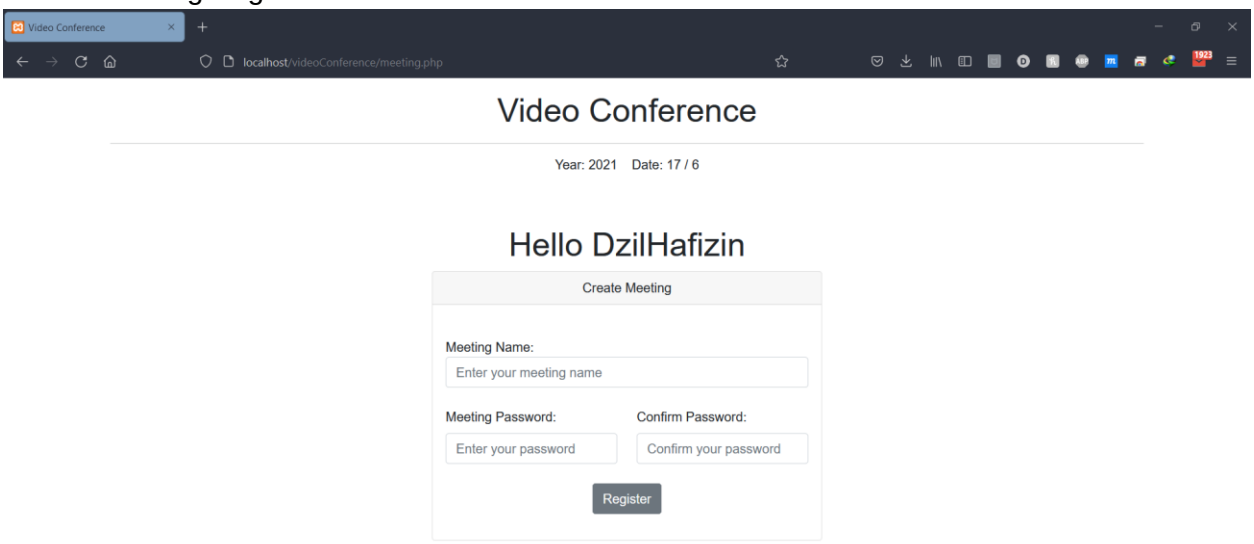
This is the page to enter the meeting by log in to the existing account.

2. Register Page



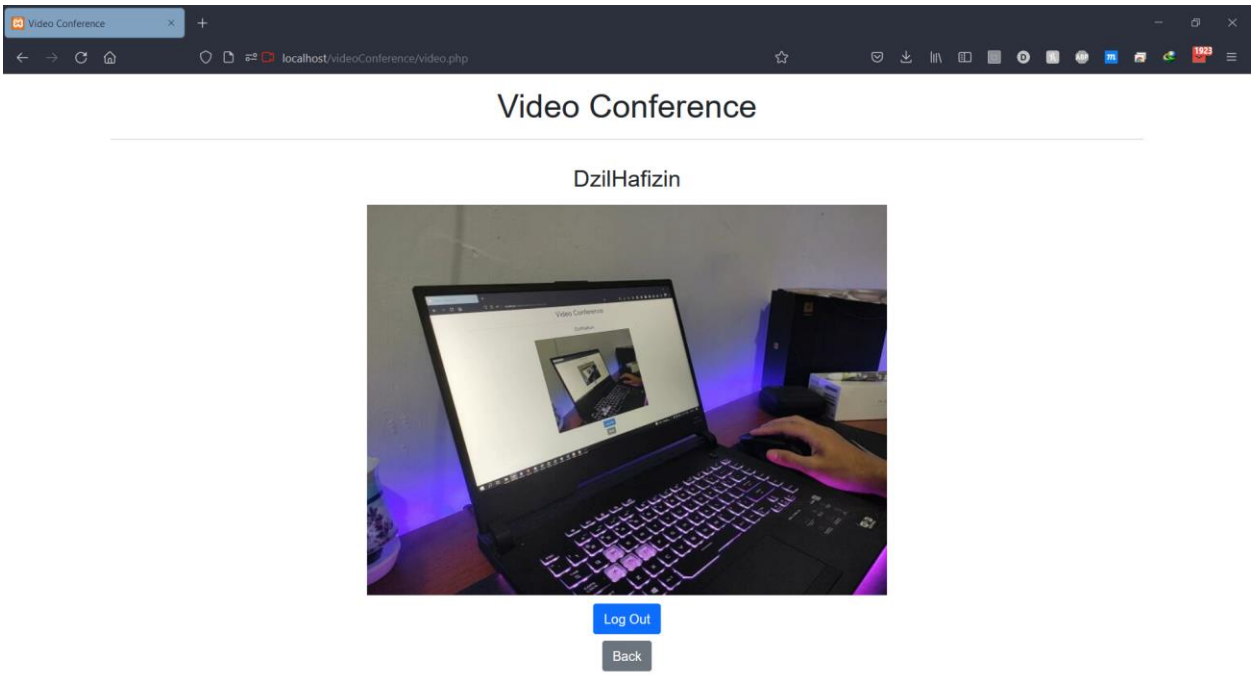
This page is to register new users by providing the required information.

Create Meeting Page



Here is the create meeting page where the user must provide the meeting name, and the password for the meeting.

3. Video Page



The video page is a meeting page where it can access your camera and display it into the page.

Implementation of Slim API:

1. Post-Method:
    a. api/login

```php
$app->post('/login', function (Request $request, Response $response, array $args) {
    $username = $_POST["username"];
    $pass = $_POST['pass'];
    // $putParams = $request->getParsedBody();
    // $username = $putParams['username'];
    // $pass = $putParams['pass'];

    session_start();

    $sql = "SELECT id, username FROM user WHERE username = (:username) AND pass = (:pass)";

    try {
        $db = new DB();
        $conn = $db->connect();

        $stmt = $conn->prepare($sql);
        $stmt -> bindValue(':username', $username);
        $stmt -> bindValue(':pass', $pass);

        $stmt->execute();
        $id = $stmt->fetch();
        $db = null;

        $_SESSION["id"] = $id;
        $_SESSION["username"] = $username;
        echo json_encode($id);
    } catch(PDOException $e) {
        $error = array(
            "status" => "fail"
        );
        echo json_encode($error);
    }
});
```

b. api/addUser

```php
$app->post('/addUser', function (Request $request, Response $response, array $args) {

    $fullname = $_POST["fullname"];
    $username = $_POST["username"];
    $email = $_POST['email'];
    $phone = $_POST['phone'];
    $pass = $_POST['pass'];
    $gender = $_POST['gender'];

    $sql = "INSERT INTO user (fullname, username, email, phone, pass, gender) VALUE (:fullname, :username, :email, :phone, :pass, :gender)";

    try {
        $db = new DB();
        $conn = $db->connect();

        $stmt = $conn->prepare($sql);
        $stmt -> bindValue(':fullname', $fullname);
        $stmt -> bindValue(':username', $username);
        $stmt -> bindValue(':email', $email);
        $stmt -> bindValue(':phone', $phone);
        $stmt -> bindValue(':pass', $pass);
        $stmt -> bindValue(':gender', $gender);

        $result = $stmt->execute();
        $db = null;

        echo json_encode($result);

    } catch(PDOException $e) {
        $error = array(
            "status" => "fail"
        );
        echo json_encode($error);
    }
});
```

c. api/addMeeting

```php
$app->post('/addMeeting', function (Request $request, Response $response, array $args) {
    $username = $_POST["username"];
    $meetingname = $_POST['meetingname'];
    $pass = $_POST['pass'];
    // $putParams = $request->getParsedBody();
    // $username = $putParams['username'];
    // $meetingname = $putParams['meetingname'];
    // $pass = $putParams['pass'];

    session_start();

    $sql = "INSERT INTO meeting (username, meetingname, pass) VALUE (:username, :meetingname, :pass)";

    try {
        $db = new DB();
        $conn = $db->connect();

        $stmt = $conn->prepare($sql);
        $stmt -> bindValue(':username', $username);
        $stmt -> bindValue(':meetingname', $meetingname);
        $stmt -> bindValue(':pass', $pass);

        $result = $stmt->execute();
        $db = null;

        echo json_encode($result);
    } catch(PDOException $e) {
        $error = array(
            "status" => "fail"
        );
        echo json_encode($error);
    }
});
```

2. Get-Method:
    a. api/getUser

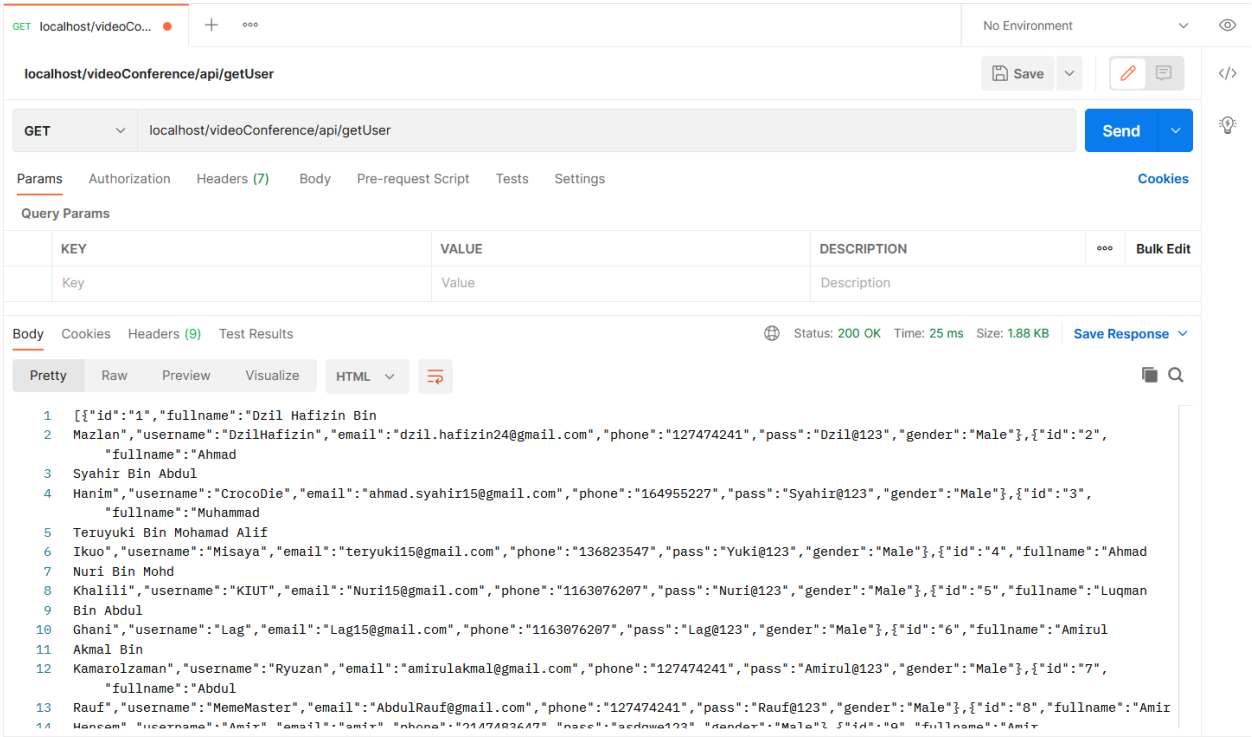
```php
$app->get('/getUser', function (Request $request, Response $response, array $args) {
    $sql = "SELECT * FROM user";

    try {
        $db = new DB();
        $conn = $db->connect();

        $stmt = $conn->query($sql);
        $user = $stmt->fetchAll(PDO::FETCH_OBJ);

        $db = null;

        // $response->getBody()->write(json_encode($user));
        // return $response;
        echo json_encode($user);

    } catch(PDOException $e) {
        $error = array(
            "status" => "fail"
        );
        echo json_encode($error);
    }
});
```

b. api/getUser/{id}

```php
$app->get('/getUser/{id}', function (Request $request, Response $response, array $args) {
    $id = $args['id'];
    $sql = "SELECT * FROM user WHERE id = $id";

    try {
        $db = new DB();
        $conn = $db->connect();

        $stmt = $conn->query($sql);
        $user = $stmt->fetchAll(PDO::FETCH_OBJ);

        $db = null;

        // $response->getBody()->write(json_encode($user));
        // return $response;
        echo json_encode($user);

    } catch(PDOException $e) {
        $error = array(
            "status" => "fail"
        );
        echo json_encode($error);
    }
});
```

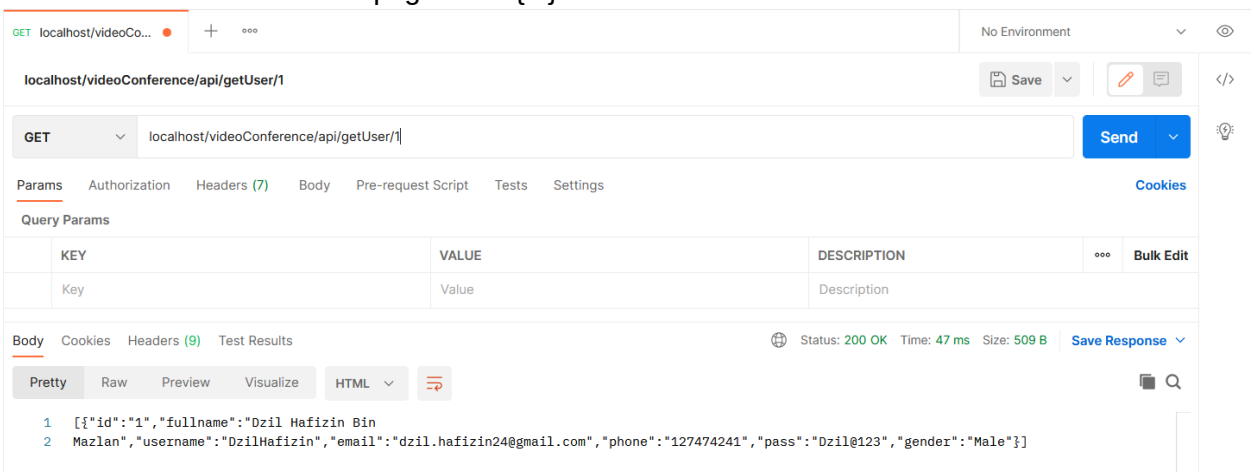
Postman:

1. localhost/videoConference/api/getUser



This is the get method that is being used to get all lists of Users by using the slim api. This can be checked using the postman by typing the link path to the slim api of getUser.

2. localhost/videoConference/api/getUser/{id}



This is the get method that is being used to get a single user by using the slim api. This can be checked using the postman by typing the link path to the slim api of getUser/{id}.

Database (DB Name: videoconference):

1. Table user:

	id	fullname	username	email	phone	pass	gender
Edit Copy Delete	1	Dzil Hafizin Bin Mazlan	DzilHafizin	dzil.hafizin24@gmail.com	127474241	Dzil@123	Male
Edit Copy Delete	2	Ahmad Syahir Bin Abdul Hanim	CrocoDie	ahmad.syahir15@gmail.com	164955227	Syahir@123	Male
Edit Copy Delete	3	Muhammad Teruyuki Bin Mohamad Alif Ikuo	Misaya	teryuki15@gmail.com	136823547	Yuki@123	Male
Edit Copy Delete	4	Ahmad Nuri Bin Mohd Khalili	KIUT	Nuri15@gmail.com	1163076207	Nuri@123	Male
Edit Copy Delete	5	Luqman Bin Abdul Ghani	Lag	Lag15@gmail.com	1163076207	Lag@123	Male
Edit Copy Delete	6	Amirul Akmal Bin Kamarolzaman	Ryuzan	amirulakmal@gmail.com	127474241	Amirul@123	Male
Edit Copy Delete	7	Abdul Rauf	MemeMaster	AbdulRauf@gmail.com	127474241	Rauf@123	Male
Edit Copy Delete	8	Amir Hensem	Amir	amir	2147483647	asdqwe123	Male
Edit Copy Delete	9	Amir Hensem	Amir	amir	2147483647	asdqwe123	Male
Edit Copy Delete	10	kirin	kirin	amirchelsea@gmail.com	2147483647	asdqwer	Male

Here is all the data of the user after being registered, it will store in this table user. The login also uses the data from this table to compare the username and password.

2. Table meeting:

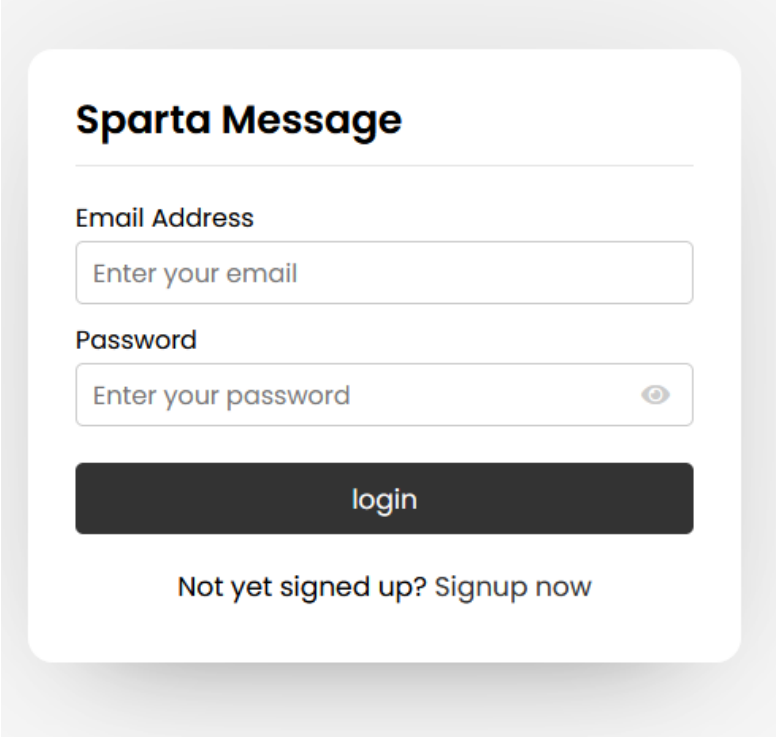
	id	username	meetingname	pass 1
Edit Copy Delete	3	CrocoDie	Usrah Agung	Alhamdulillah
Edit Copy Delete	1	DzilHafizin	Annual Meeting	Dzil123

Here is all the data of the meeting being created before start the meeting, it will store the meeting name, and created by who with a password.

Text Messaging and File Sharing

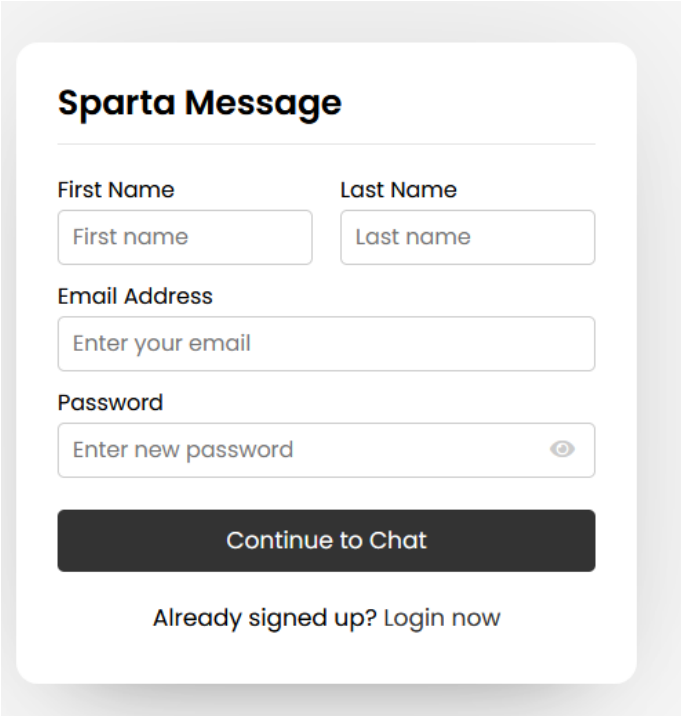
User Interface:

1. **Login Page**



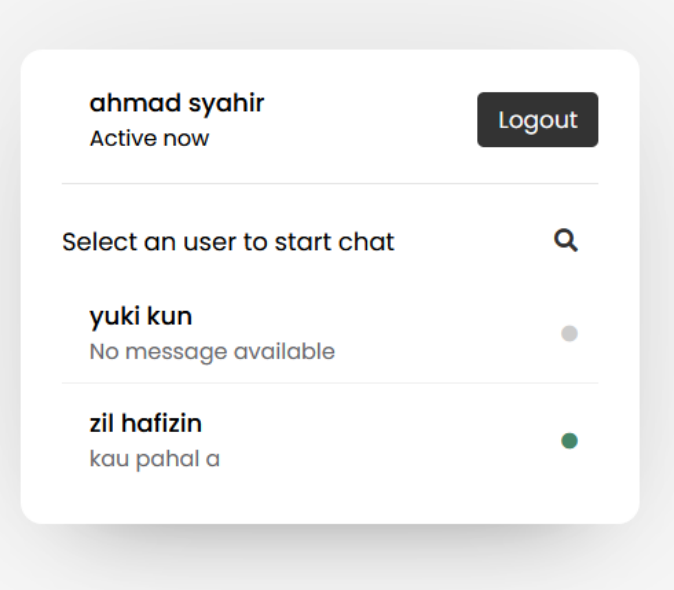
To login to the system, the user needs to input the email and the password, then by using the email and the password, the system will get the user data from the database using slim framework to return it as json. The connection and json data can be checked using Advanced Rest Client.

2. **Register Page**



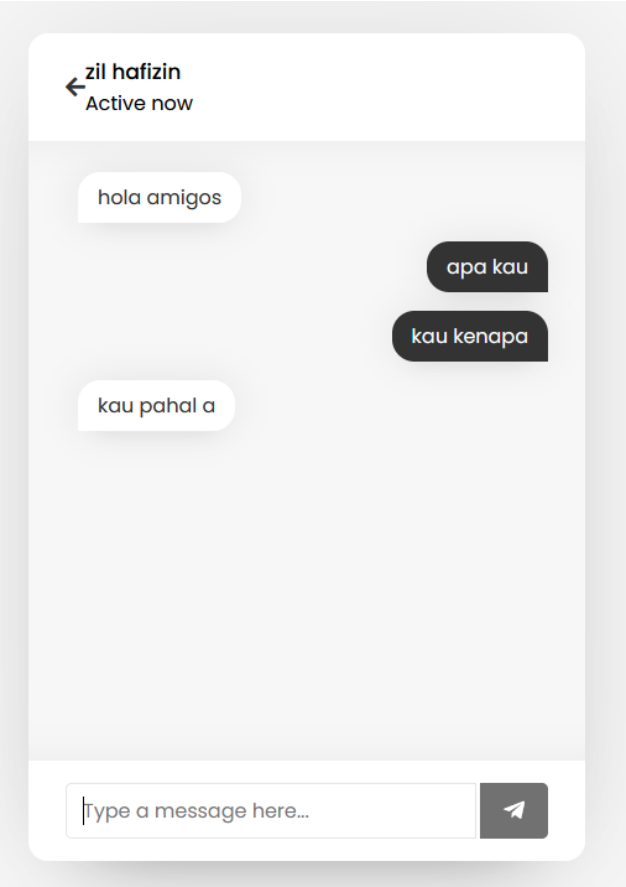
To register to the system, the user needs to input the first name, last name, email address, and password.

3. **Friends List**



This is the profile page, and users can view the friends list that are online and available to chat. Green dot represents an active user, while grey dot represents a non active user.

4. **Chat Page**

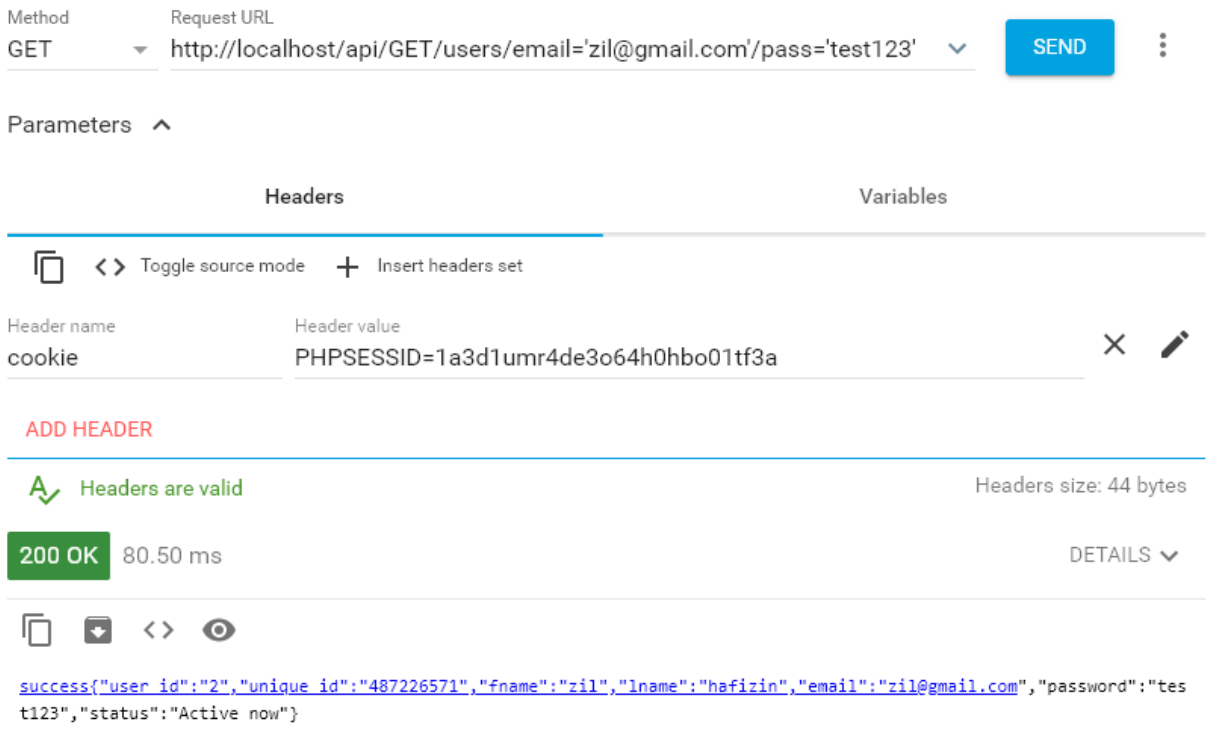


The user can send a message by entering the message on the chat box and then press enter to send to another user.

5. Advance Rest Client and Slim Implementation in Coding

```php
$app->get('/GET/users/email={email}/pass={password}', function (Request $request, Response $response, $args) {
    session_start();
    $email = $args['email'];
    $password = $args['password'];

    if(!empty($email) && !empty($password)){
        $sql = "SELECT * FROM users WHERE email = $email";
        try {
            // Get DB Object
            $db = new db();
            // Connect
            $db = $db->connect();

            $stmt = $db->query($sql);
            $user = $stmt->fetch(PDO::FETCH_OBJ);
            $user_pass = $user->password;
                $status = "Active now";
                $sql2 = "UPDATE users SET status = :status WHERE email = $email";
                $stmt = $db->prepare($sql2);
                $stmt->bindParam(':status', $status);
                $stmt->execute();
                $count = $stmt->rowCount();
                if($sql2){
                    $_SESSION['unique_id'] = $user->unique_id;
                    echo "success";
                }else{
                    echo "Something went wrong. Please try again!";
                }
            $db = null;
            echo json_encode($user);
        } catch (PDOException $e) {
            $data = array(
                "status" => "fail"
            );
            echo json_encode($user->unique_id);
        }
    }
```

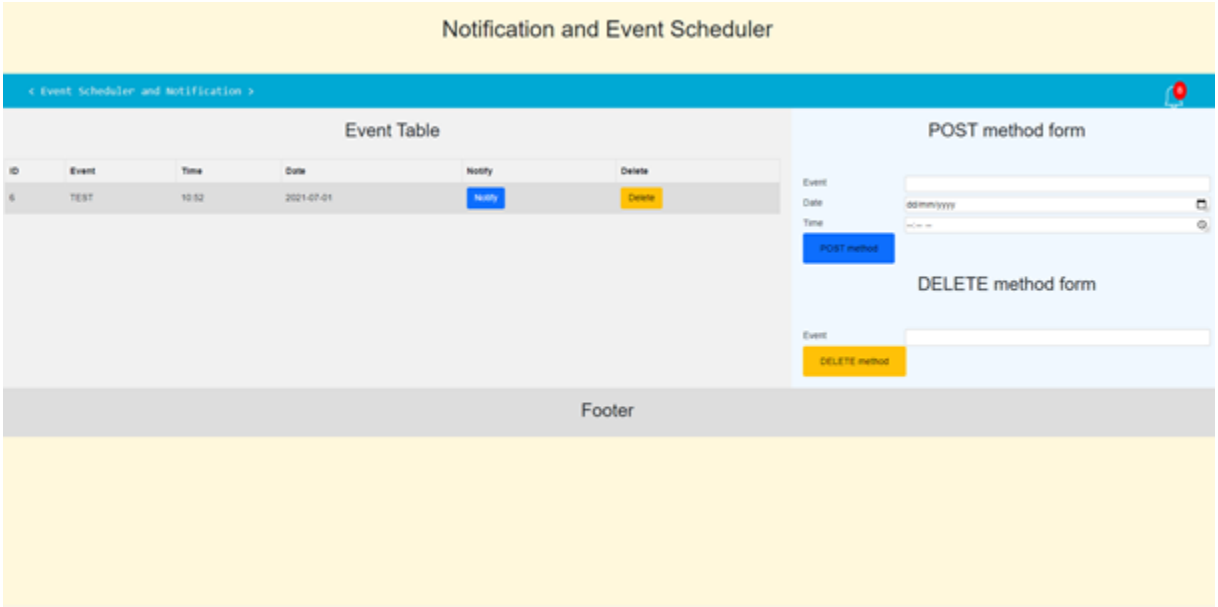
By using the slim framework, we can get the email and the password of the user using get method and using args to pass the value.



The json will be displayed on the Advance Rest Client and show the connection to the server side is successful.

Event Scheduler and Notification

User Interface



Feature 1- Getting Data From Database

+ Options

				id	link
☐	Edit	Copy	Delete	8	https://media.istockphoto.com/photos/owl-watches-i...
☐	Edit	Copy	Delete	9	https://media.istockphoto.com/photos/owl-watches-i...
☐	Edit	Copy	Delete	10	https://images.unsplash.com/photo-1493540447904-49...
☐	Edit	Copy	Delete	11	https://images.unsplash.com/photo-1462007895615-c8...
☐	Edit	Copy	Delete	12	https://images.unsplash.com/photo-1462007895615-c8...

For the image, they are getting data from the table with named “assigment3” which contains id and the image link.

```php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Methods: GET, POST, OPTIONS, DELETE");

class db{
    // Properties
    private $host = 'localhost';
    private $user = 'root';
    private $password = '';
    private $dbname = 'assignment3';

    // Connect
    public function connect(){
        $mysql_connect_str = "mysql:host=$this->host;dbname=$this->dbname";
        $dbConnection = new PDO($mysql_connect_str, $this->user, $this->password);
        $dbConnection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $dbConnection;
    }
}
```

Figure 3: The configuration of the database in the localhost in the db.php

Feature 2 - Show event that added

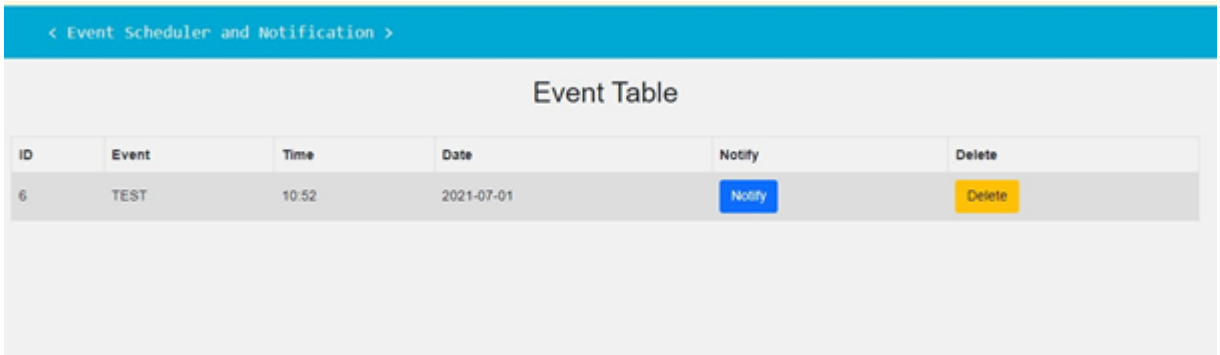


Image 1

```php
$app->get('/events', function ($request,  $response, $args) {
    // $response->getBody()->write("this will return all users");
    // return $response;

    $sql = "SELECT * FROM event";
    try{
        // Get DB Object
        $db = new db();
        // Connect
        $db = $db->connect();

        $stmt = $db->query($sql);
        $user = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        echo json_encode($user);
    }catch (PDOException $e) {
        $data = array(
            "status" => "fail"
        );
        echo json_encode($data);
    }
});

$app->post('/addEvent',function($request, $response, $args){
    //connect to database
    $db = new db();
    $db = $db->connect();
    //sql insert etc....
$event = $_POST["event"];
$time = $_POST["time"];
$date = $_POST["date"];
```

Figure 5: In the index.php to get data from database and pass to the JavaScript.

Feature 3 - Add event

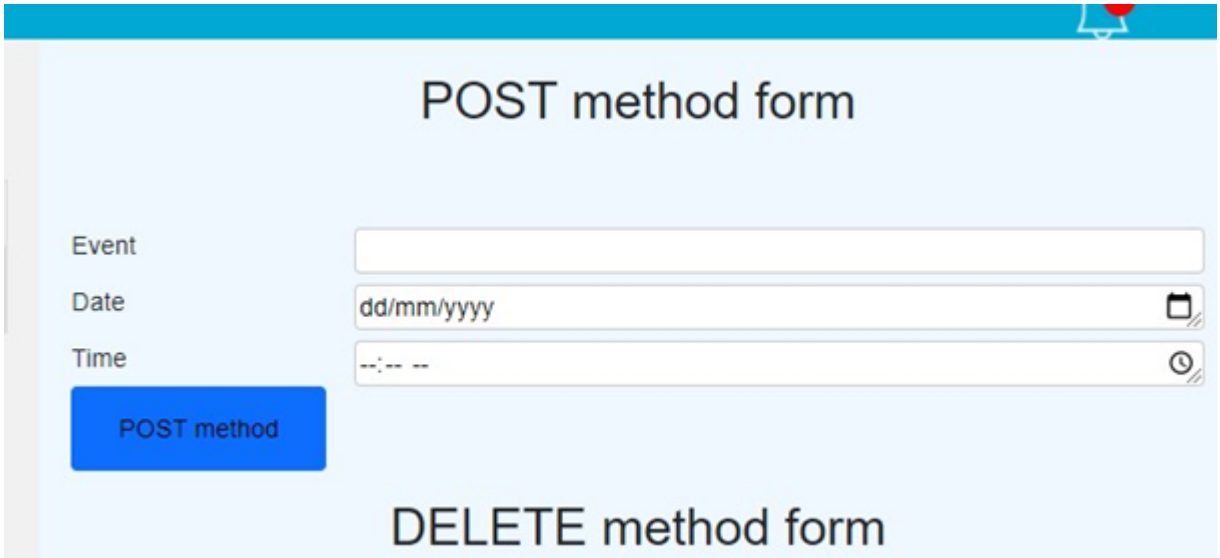


Figure 7: Need to enter the detail of the event before add.

```
    try {
        $sql = "INSERT INTO event (event_name, date, time) VALUES ('$event','$date','$time')";
        $db = new db();
        // Connect
        $db = $db->connect();
        $stmt = $db->prepare($sql);
        $stmt->bindValue(':event_name', $event);
        $stmt->bindValue(':date', $date);
        $stmt->bindValue(':time', $time);
        $stmt->execute();
        $count = $stmt->rowCount();
        $db = null;

        $data = array(
            "status" => "success",
            "rowcount" =>$count
        );
        echo json_encode($data);
    } catch (PDOException $e) {
        $data = array(
            "status" => "fail"
        );
        echo json_encode($data);
    }


    //return the status insert berjaya/tidak, as a response
    // $response->getBody()->write("this operation will insert user ti database table");
    // return $response;
});
```

Figure 8: The form that get data from user in the index.html

```javascript
const date = [];

function loadEvents(){
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'slim/events', true);

    xhr.onload = function(){
      if(this.status == 200){
        var events = JSON.parse(this.responseText);

        var output = '';
        output = '<tr>' +
            '<th>ID</th>' +
            '<th>Event</th>' +
            '<th>Time</th>' +
            '<th>Date</th>' +
            '<th>Notify</th>' +
            '<th>Delete</th>' +
            '</tr>';

        for(var i in events){
            date[i] = events[i].date;
            event_name[i] = events[i].event_name;
            time[i] = events[i].time;
            id[i] = events[i].id;

          output += '<tr>' +
            '<td>' + events[i].id +'</td>' +
            '<td>' + events[i].event_name +'</td>' +
            '<td>' + events[i].time +'</td>' +
            '<td>' + events[i].date +'</td>' +
            '<td><button type="button" class="btn btn-primary" onclick="myNotify(' + i + ')">Notify</button></td>' +
            '<td><button type="button" class="btn btn-warning" onclick="myDelete(' + i + ')">Delete</button></td>' +
            '</tr>';
        }

        document.getElementById('tableGET').innerHTML = output;
      }
    }

    xhr.send();
}
```

Figure 9: In the nuri.js, get the data from the index.html and pass the data to the Slim framework (Index.php)

```php
});

$app->get('/events', function ($request,  $response, $args) {
    // $response->getBody()->write("this will return all users");
    // return $response;

    $sql = "SELECT * FROM event";
    try{
        // Get DB Object
        $db = new db();
        // Connect
        $db = $db->connect();

        $stmt = $db->query($sql);
        $user = $stmt->fetchAll(PDO::FETCH_OBJ);
        $db = null;
        echo json_encode($user);
    }catch (PDOException $e) {
        $data = array(
            "status" => "fail"
        );
        echo json_encode($data);
    }
});

$app->post('/addEvent',function($request, $response, $args){
    //connect to database
    $db = new db();
    $db = $db->connect();
    //sql insert etc....
$event = $_POST["event"];
$time = $_POST["time"];
$date = $_POST["date"];

// $inputJSON = file_get_contents('php://input');
// $input = json_decode($inputJSON, TRUE);
// $name = $input["name"];


    try {
        $sql = "INSERT INTO event (event_name, date, time) VALUES ('$event','$date','$time')";
        $db = new db();
        // Connect
        $db = $db->connect();
        $stmt = $db->prepare($sql);
        $stmt->bindValue(':event_name', $event);
        $stmt->bindValue(':date', $date);
        $stmt->bindValue(':time', $time);
```

Figure 10: In the index.php, the system will passing the data to the database for store the data.

Feature 4- Delete Event

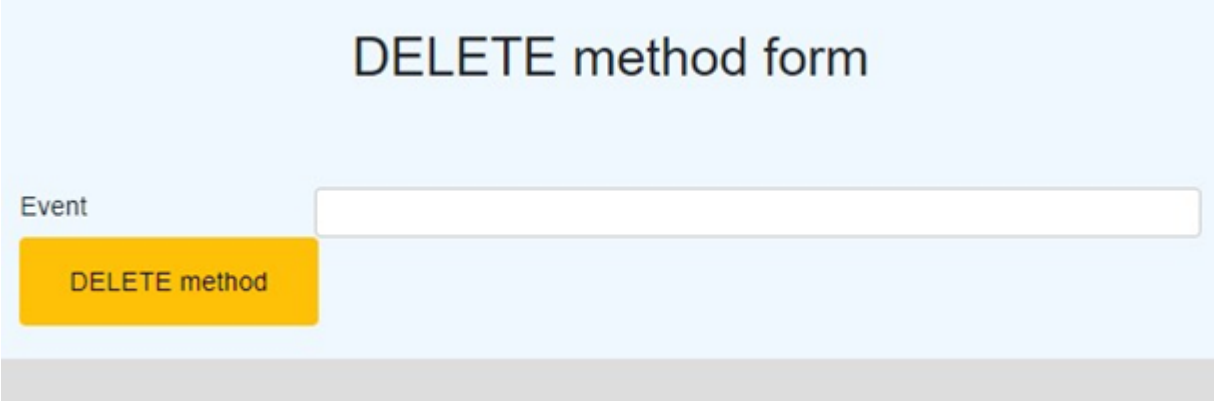


Figure 11: The form in the index.html

```
            <div class="col-75"><input type="time" name="time"> <br></div>
            <p style="text-align: center;"><div class="btn btn-primary" style="float: none;"><input type="submit" value="POST method"></div></
        </form>
        <form id="deleteForm">
            <h2 style="text-align: center;">DELETE method form</h2><br>
            <br>
            <div class="col-25"><label for="event">Event </label></div>
            <div class="col-75"><input type="text" name="event"> <br></div>
            <div class="btn btn-warning" style="float: center;"><input type="submit" value="DELETE method"></div>
        </form>
    </div>
  </div>
  <div class="footer">
      <h2>Footer</h2>
  </div>
  <script src="muri.js"></script>
/body>
/html>
```

Figure 12: The form in the index.html to enter event name to be delete.

```javascript
deleteForm.onsubmit = async (e) => {
    e.preventDefault();

    let response = await fetch('slim/index.php/deleteEvent', {
      method: 'POST',
      body: new FormData(deleteForm)
    });

    let result = await response.json();

    alert('Event deleted');
    loadEvents();
};

var box = document.getElementById('box');
    var down = false;
    function toggleNotifi() {
        if (down) {
            box.style.height = '0px';
            box.style.opacity = 0;
            down = false;
        } else {
            box.style.height = '1000px';
            box.style.opacity = 1;
            down = true;
        }
}

function myNotify(i){
        noti += '<br><div class="text" style="border: 1px solid black; width: 100%; margin: 0">' +
                '<h9>' + date[i] + '  |  ' + time[i] + '  |  ' + event_name[i] + '</h9>' +
            '</div>';

    document.getElementById("NotifyT").innerHTML = noti;
```

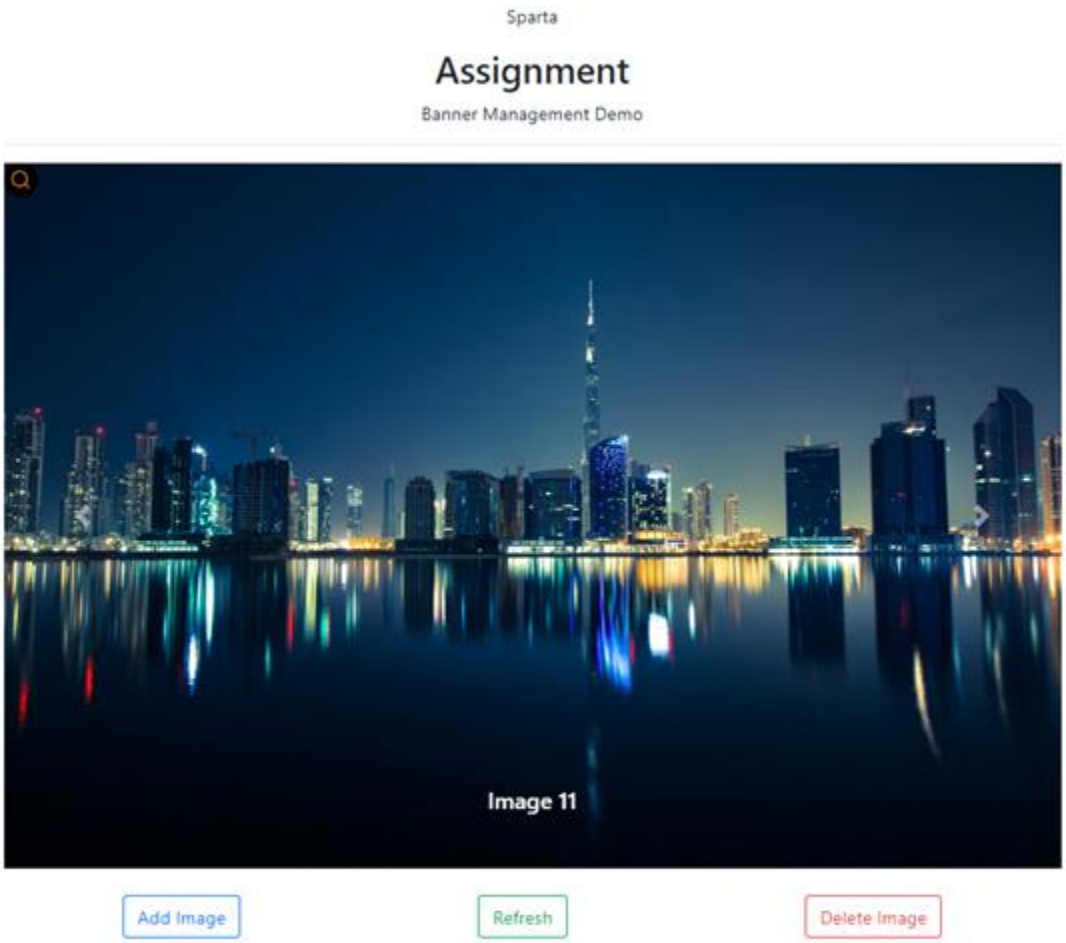
Figure 13: In the script.js to pass the data to the database.

```php
$app->post('/deleteEvent',function($request, $response, $args){
    //connect to database
    $db = new db();
    $db = $db->connect();
    //sql insert etc....
$event = $_POST["event"];


    try {
        $sql = "DELETE FROM event WHERE event_name='$event'";
        $db = new db();
        // Connect
        $db = $db->connect();
        $stmt = $db->prepare($sql);
        $stmt->bindValue(':event_name', $event);

        $stmt->execute();
        $count = $stmt->rowCount();
        $db = null;

        $data = array(
            "status" => "success",
            "rowcount" =>$count,

        );

        echo json_encode($data);
    } catch (PDOException $e) {
        $data = array(
            "status" => "fail"
        );
        echo json_encode($data);
    }

});

$app->delete('/deleteEvent/{id}',function($request, $response, $args){
    $id = $args['id'];
    //connect to database
```

Figure 14: In the index.php, the system will run the script to delete the event in the database.

Banner Management

Feature 1: Get All data from Database.



Whole Screen

The feature to display the data from the database is by using Carousel from Bootstrap 4 function. And the data is got from the database by using the Slim Framework. And, the image will rotate automatically.

In this page shows all the image that stored in the database and show the Image and the imageID. This is for the user to easily to know which photo they are watching with.

+ Options

←T→				id	link
☐	Edit	Copy	Delete	8	https://media.istockphoto.com/photos/owl-watches-i...
☐	Edit	Copy	Delete	9	https://media.istockphoto.com/photos/owl-watches-i...
☐	Edit	Copy	Delete	10	https://images.unsplash.com/photo-1493540447904-49...
☐	Edit	Copy	Delete	11	https://images.unsplash.com/photo-1462007895615-c8...
☐	Edit	Copy	Delete	12	https://images.unsplash.com/photo-1462007895615-c8...

Database Structure

For the image, they are getting data from the table with named “imagelink” which contains id and the image link.

```php
db.php
<?php
header('Access-Control-Allow-Origin: *');
header("Access-Control-Allow-Methods: GET, POST, OPTIONS, DELETE");

    class db{
        // Properties

        // // database
        private $host = 'localhost';
        private $user = 'root';
        private $password = '';
        private $dbname = 'weba3';

        // Connect
        public function connect(){
            $mysql_connect_str = "mysql:host=$this->host;dbname=$this->dbname";
            $dbConnection = new PDO($mysql_connect_str, $this->user, $this->password);
            $dbConnection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            return $dbConnection;
        }
    }
?>
```

The configuration of the database in the localhost in the db.php

```html
<div id="showBanner" class="carousel slide" data-ride="carousel">

</div>
```

The div tag to display the image

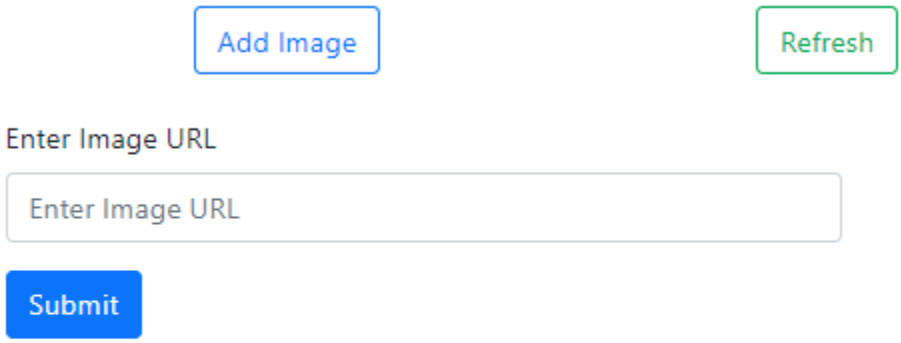
```javascript
loadPicture();

function loadPicture() {
  window.onload = function () {
    $.ajax({
      type: "GET",
      url: "src/getBannerData",
      dataType: "json",
      success: function (bannerlist, status, xhr) {
        console.log(bannerlist);

        var display = " ";
        var indicator = "";
        var banners = "";
        $.each(bannerlist, function (index, value) {

          if (index == 0) {
            indicator +=
              '  <li data-target="#showBanner" data-slide-to="0" class="active"></li>';
            banners +=
              '  <div class="carousel-item active">  <img class="d-block w-100" src="' +
              value.bannerURL +
              '" alt="img' +
              value.id +
              '">  <div class="carousel-caption d-none d-md-block"> <h5> Image '+ value.id +'</h5> </div></div>';
          } else {
            indicator += '<li data-target="#showBanner" data-slide-to="'+index+'"></li>';
            banners +=
              '<div class="carousel-item"><img class="d-block w-100" src="' +
              value.bannerURL +
              '" alt="img' +
              value.id +
              '">'+
              '<div class="carousel-caption d-none d-md-block">'+
              '<h5> Image '+ value.id +'</h5>'+
            '</div></div>';
          }
        });
        console.log(banners);
        display =
          /*' <ol class="carousel-indicators">' +
          indicator +
          ' </ol>'+ */
          '<div class="carousel-inner">' +
          banners +
          '</div>'+
          '<a class="carousel-control-prev" href="#showBanner" role="button" data-slide="prev">'+
          '<span class="carousel-control-prev-icon" aria-hidden="true">'+
          '</span><span class="sr-only">Previous</span></a> '+
          '<a class="carousel-control-next" href="#showBanner" role="button" data-slide="next">  '+
          '<span class="carousel-control-next-icon" aria-hidden="true"></span> '+
          '<span class="sr-only">Next</span>  </a> ';
        $("#showBanner").append(display);
      },
      error: function (xhr) {
        console.log(xhr);
        console.log("ajaxx");
      },
    });
  };
```

: in the JavaScript with named script.js and get the and arrange the data to show in the html.

```php
$app->get('/getBannerData', function (Request $request, Response $response, array $args) {
    class Banner
    {
        public $id = "";
        public $bannerURL = "";
    }

$data = array();

try{
    $db = new db();
    $db = $db->connect();

    $stmt = $db->query("SELECT * FROM imagelink");
    while ($row = $stmt->fetch()){
        //create an object/instance user of class Book
        $banner = new Banner();
        //populate the data/properties of object book
    $banner->bannerURL = $row['link'];
    $banner->id = $row['id'];
    array_push($data,$banner);
    }
    }catch(PDOException $e){
        echo "Connection failed: " . $e->getMessage();
    }

    echo json_encode($data);
});
```

In the index.php to get data from database and pass to the JavaScript.

Feature 2: Add image.



The image URL to required to user to enter the image URL to add the banner.

```
<div class="container" style="display: none;" id="add">
  <form>
    <div class="form-group">
      <label for="url">Enter Image URL</label>
      <input type="text" class="form-control" id="url" style="width: 50%;" placeholder="Enter Image URL">
    </div>
    <button type="submit" class="btn btn-primary" onclick="addBanner()">Submit</button>
  </form>
</div>
```

The form that get data from user in the index.html

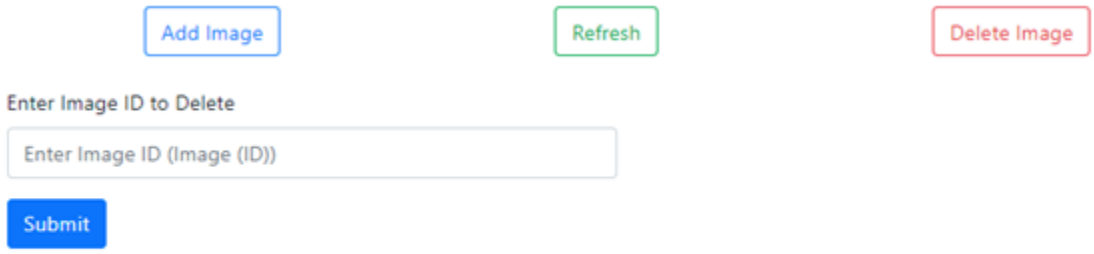
```javascript
function openAdd(){
  document.getElementById("add").style.display="block";
}

function addBanner(){
  var imageURL = $("#url").val()
  console.log(imageURL);
  $.ajax(
    {
      type: "POST",
      url: "src/addBannerData",
      data: {
          "imageURL": imageURL,
      },
      dataType: "json",
      success: function (data, status, xhr) {
        alert("Added successful")
        location.reload()
      },
      error: function (data, status, xhr) {
        alert(xhr)
      }
    }
  )
}
```

In the script.js, get the data from the index.html and pass the data to the Slim framework (Index.php)

```php
$app->post('/addBannerData', function (Request $request, Response $response, array $args){
    $link = $_POST["imageURL"];

    try{
        $sql = "INSERT INTO imagelink (link, id) VALUES (:link, NULL)";
        $db = new db();

        $db = $db->connect();
        $stmt = $db->prepare($sql);
        $stmt->bindValue(':link', $link);
        $stmt->execute();
        $count = $stmt->rowCount();
        $db = null;

        $data = array(
            "status" => "success",
            "rowcount" => $count
        );

            echo json_encode($data);
        } catch (PDOException $e) {
        $data = array(
            "status" => "fail"
        );

            echo json_encode($data);
        }
});
```

In the index.php, the system will passing the data to the database for store the data.

Feature 3: Remove image.



: The form in the index.html

```
<div class="container" style="display: none;"  id="delete">
  <form>
    <div class="form-group">
      <label for="imageID">Enter Image ID to Delete</label>
      <input type="text" class="form-control" id="imageID" style="width: 50%;" placeholder="Enter Image ID (Im
    </div>
    <button type="submit" class="btn btn-primary" onclick="deleteBanner()">Submit</button>
  </form>
</div>
```

The form in the index.html to get the which image to remove. User will enter the id to select the data.

```javascript
function openDelete(){
  document.getElementById("delete").style.display="block";
}

function deleteBanner(){
  var imageID = $("#imageID").val()
  console.log(imageID);
  $.ajax(
    {
      type: "DELETE",
      url: "src/deleteBannerData/" + imageID,
      dataType: "json",
      success: function(data, status, xhr){
        alert("Deleted ")
        location.reload()
      },
      error: function (data, status, xhr) {
          alert(xhr)
      }
    }
  )
}
```

In the script.js to pass the data to the database.

```php
$app->delete('/deleteBannerData/{id}', function (Request $request, Response $response, array $args){
    $id = $args['id'];
    $sql = "DELETE FROM imagelink WHERE id = $id";

    try{
        $sql = "DELETE FROM imagelink WHERE id = $id";
        $db = new db();

        $db = $db->connect();
        $stmt = $db->prepare($sql);
        $stmt -> execute();
        $count = $stmt->rowCount();

        $db = null;
        $data = array(
            "rowAffected" => $count,
            "status" => "success"
        );

        echo json_encode($data);
    }catch (PDOException $e) {
        $data = array(
            "status" => "fail"
        ); echo json_encode($data);
    }
});
$app->run();
```

In the index.php, the system will run the script to delete the id in the database.

[{"id":"1","bannerURL":"https:\/\/wallpapercave.com\/wp\/wp1875138.jpg"},
{"id":"3","bannerURL":"https:\/\/wallpapercave.com\/wp\/9Iql5du.jpg"},
{"id":"4","bannerURL":"https:\/\/wallpapercave.com\/wp\/wp2148360.jpg"},
{"id":"6","bannerURL":"https:\/\/images.wallpaperscraft.com\/image\/planet_craters_space_light_61032_1920x1080.jpg"}]

Json FIle for GET method

Method	*URL*
GET	Get all image data from the database https://spartateam.tk/assignment4/banner/api/getBannerData
POST	Post URL data https://spartateam.tk/assignment4/banner/api/addBannerData
DELETE	https://spartateam.tk/assignment4/banner/api/deleteBannerData