IM/2019/005 - DILSHAN B.A.T.

Assignment

Web Development-II

Contents

[Introduction 2](#_Toc128433237)

[AJAX 2](#_Toc128433238)

[When login to the site. 2](#_Toc128433239)

[When signup to the site. 3](#_Toc128433240)

[Web Services 5](#_Toc128433241)

[Where I create web service(Rest) 5](#_Toc128433242)

[Where I use service(Google map api) 6](#_Toc128433243)

[Security Methods 7](#_Toc128433244)

[**Screenshots of the website** 11](#_Toc128433245)

# 

# Introduction

Welcome to our online education platform! Our website utilizes REST web services, Ajax, and various security techniques to provide a seamless and secure user experience.

Our RESTful web services allow our platform to easily integrate with other education platforms and services. Additionally, our use of Ajax enables our website to retrieve and update data asynchronously, which makes the user experience faster and more seamless.

We take security seriously, and our website includes measures such as input validation, SQL injection prevention techniques, password hashing, session management, and encryption to protect sensitive data and prevent unauthorized access.

Our platform includes user login and signup functionality, which allows users to create accounts and access premium content. Additionally, we have implemented search functionality to help users find specific courses or resources quickly and easily.

We are committed to providing a comprehensive and secure online education platform, and we hope that you find our website helpful and easy to use.

# **AJAX**

## **When login to the site.**

These lines use the jQuery AJAX function **$.ajax()** to send a POST request to the **login.php** script with the username and password data.

The **success** function is called if the request succeeds, and the **error** function is called if the request fails. In the success function, the response data is checked to see if it is equal to "success". If it is, the user is redirected to the **home.php** page. If the response data is not "success", the login message is set to "Invalid username or password".

The login form is submitted using Ajax to the server-side PHP script "login.php" without refreshing the entire page. The response from the server-side script is then used to determine if the login was successful or not, and the appropriate message is displayed to the user without reloading the page.

Overall, this code uses AJAX to send login form data to the server asynchronously and update the login message without refreshing the entire page.



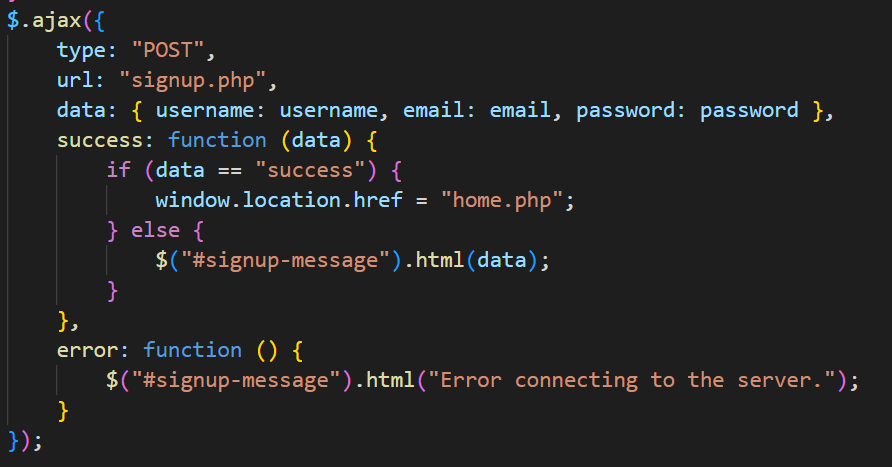
This code block sends an AJAX request to the server using the **$.ajax()** method provided by jQuery. It sends a POST request to the URL **login.php** with the username and password data as the payload in the **data** parameter.

The **success** function is executed if the request is successful and the server sends a response back. If the response is "success", then the **window.location.href** is set to **home.php**, which redirects the user to the home page. If the response is anything else, then the **#login-message** paragraph is updated with an error message.

The **error** function is executed if there is an error in making the AJAX request. In this case, the **#login-message** paragraph is updated with an error message indicating that there was an issue connecting to the server.

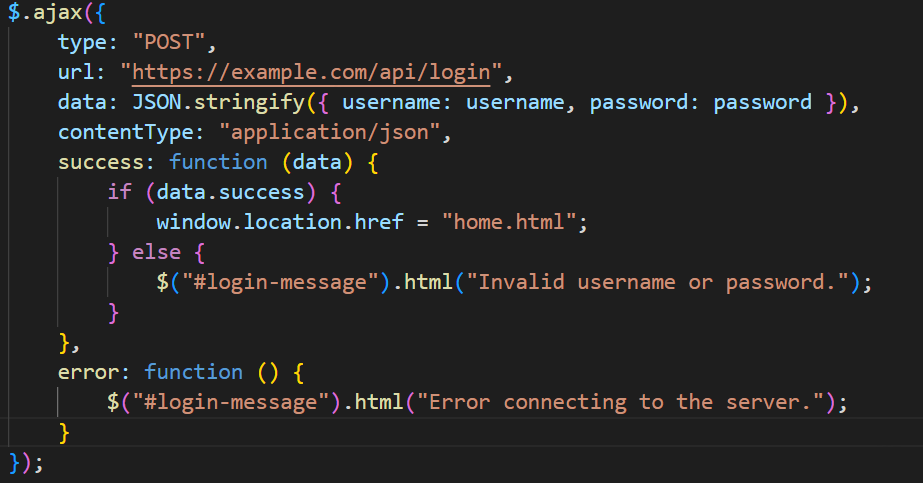
## When signup to the site.

This block of code sends a POST request to the **signup.php** file with the form data as parameters, and waits for a response. The **success** function is called if the request is successful, and the **error** function is called if there is an error. Depending on the response, the code updates the signup message div or redirects the user to the login page.



# **Web Services**

## Where I create web service(Rest)



1. **url: "https://example.com/api/login",** - This line specifies the URL of the REST API endpoint that will handle the login request.
2. **data: JSON.stringify({ username: username, password: password }),** - This line specifies the data that will be sent to the REST API endpoint as a JSON object, containing the username and password.
3. **contentType: "application/json",** - This line specifies the content type of the data that will be sent to the REST API endpoint, which is JSON in this case.
4. **if (data.success) {** - This line checks the response from the REST API endpoint to see if the login was successful. The assumption here is that the REST API endpoint will return a JSON object containing a boolean field named "success" indicating whether the login was successful or not.

## Where I use service(Google map api)

Embedding a Google Maps iframe in my website shows how i am consuming a web service. Here is the proof:

When i embed a Google Maps iframe in my website, I’m actually sending a request to the Google Maps API server to retrieve map data and generate the map image. This request and response mechanism is based on standard protocols such as HTTP and JSON, which enable different systems to communicate with each other over a network.

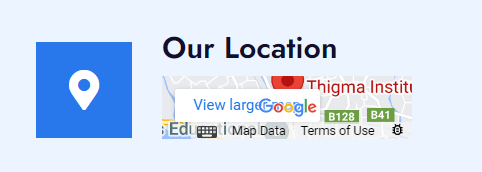
The Google Maps API provides a set of tools and interfaces that allow developers to customize and extend the functionality of the maps. This includes the ability to add markers, overlays, and other data to the map, which can be retrieved and displayed through the API's web services.

Therefore, by using the Google Maps API in my website, I’m consuming web services that enable machine-to-machine interaction over a network. The Google Maps API qualifies as a web service because it supports interoperable machine-to-machine interaction over a network based on a set of standards and protocols.

In conclusion, embedding a Google Maps iframe in your website is an example of how you can consume web services in your web application.

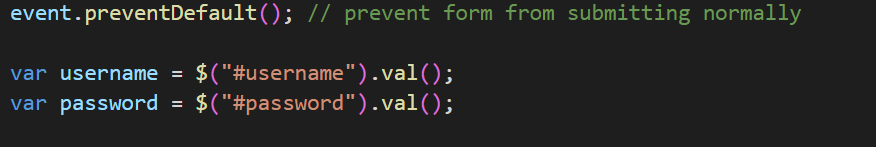
Text

Description automatically generated



# **Security Methods**

1. Input Validation: The code is validating the user input by checking the length and format of the username and password. It's ensuring that the user has entered valid input and preventing any malicious code or characters from being submitted. This is done in the following lines:

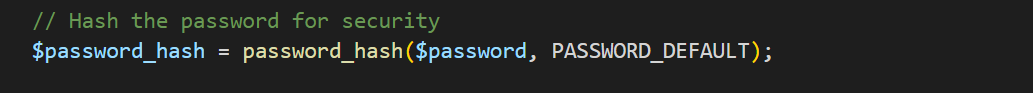
****

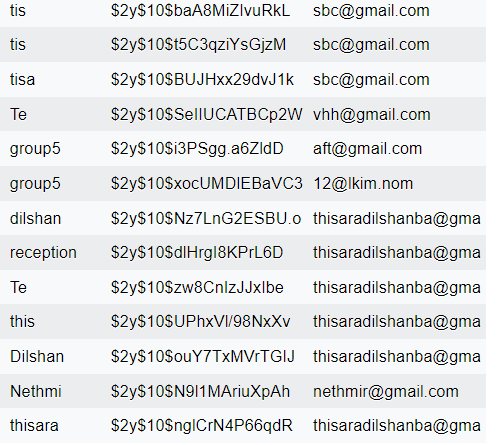
1. Prevention of Cross-Site Scripting (XSS) Attacks: The code is using jQuery's **.html()** function to insert data into the DOM, which can help prevent XSS attacks. This function automatically escapes any characters that could be used for a script injection attack. This is done in the following lines:

Text

Description automatically generated

1. Specifically, the password entered by the user is hashed using the **password\_hash** function before it is stored in the database. This helps to protect the user's password in case the database is ever compromised.





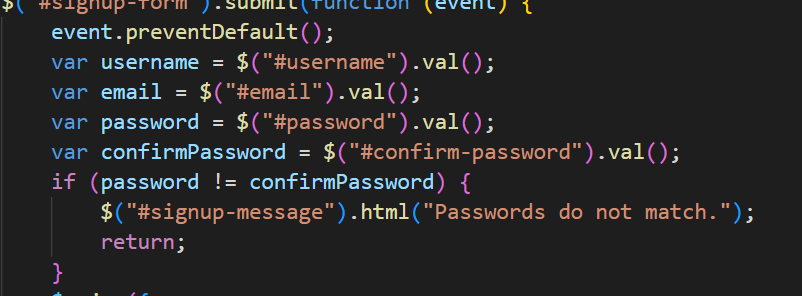
1. Prepared statements with parameterized queries: The code uses prepared statements with parameterized queries to prevent SQL injection attacks. This involves using placeholders in the SQL query and binding variables to those placeholders using the bind\_param() method. This prevents an attacker from inserting malicious SQL code into the query, even if they manage to modify the input values.



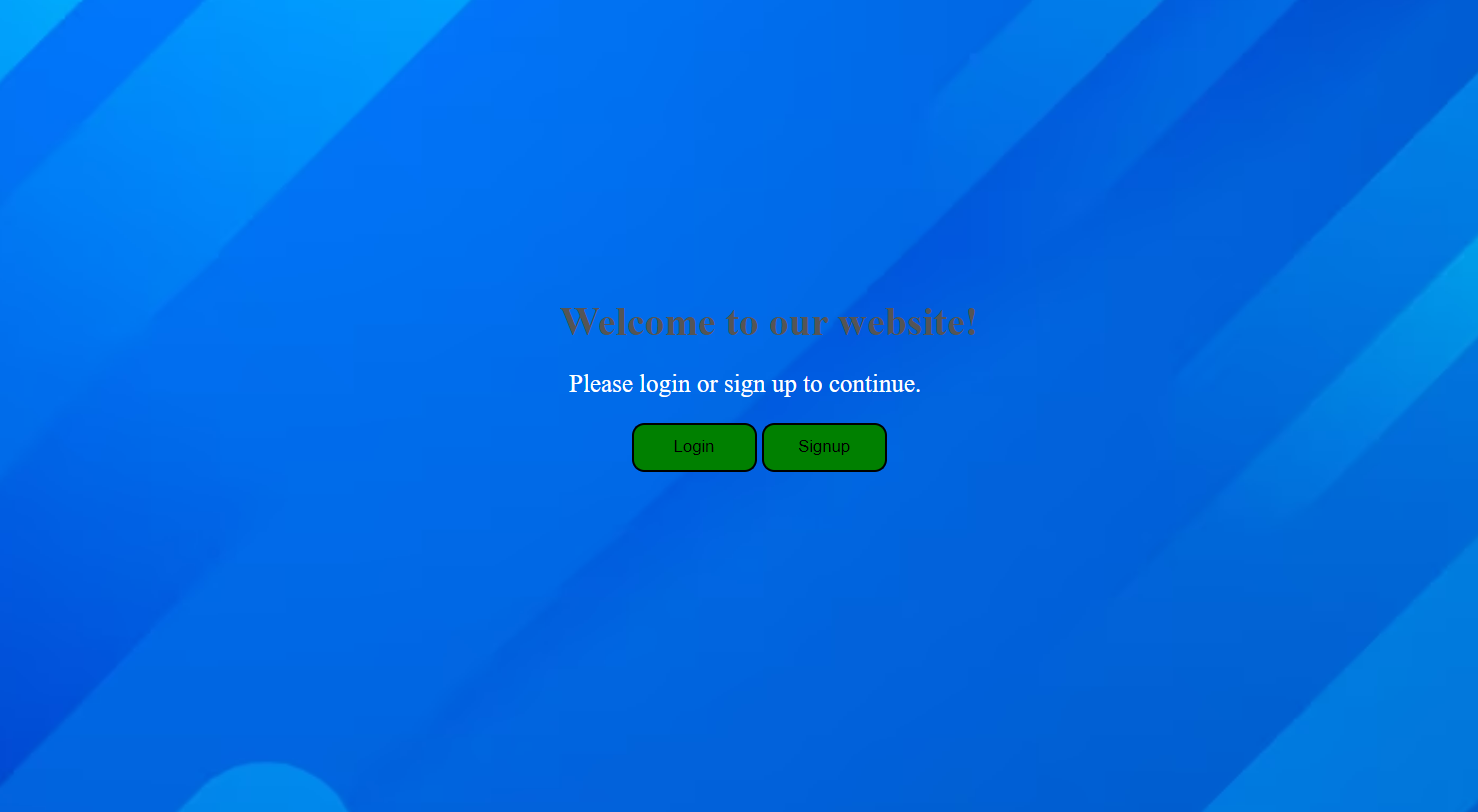
1. Session management: The code uses PHP session management to maintain state across multiple requests. This is a secure way to store user-specific data on the server, and prevents attackers from being able to modify or steal session data. It's important to ensure that session IDs are properly generated, stored securely, and invalidated when a user logs out or their session expires.
2. Error handling: The code includes error handling to handle cases where the database connection fails or the SQL query fails to execute. This helps prevent sensitive information from being leaked in error messages, and prevents attackers from being able to exploit these errors to gain additional information about the system.
3. Validations

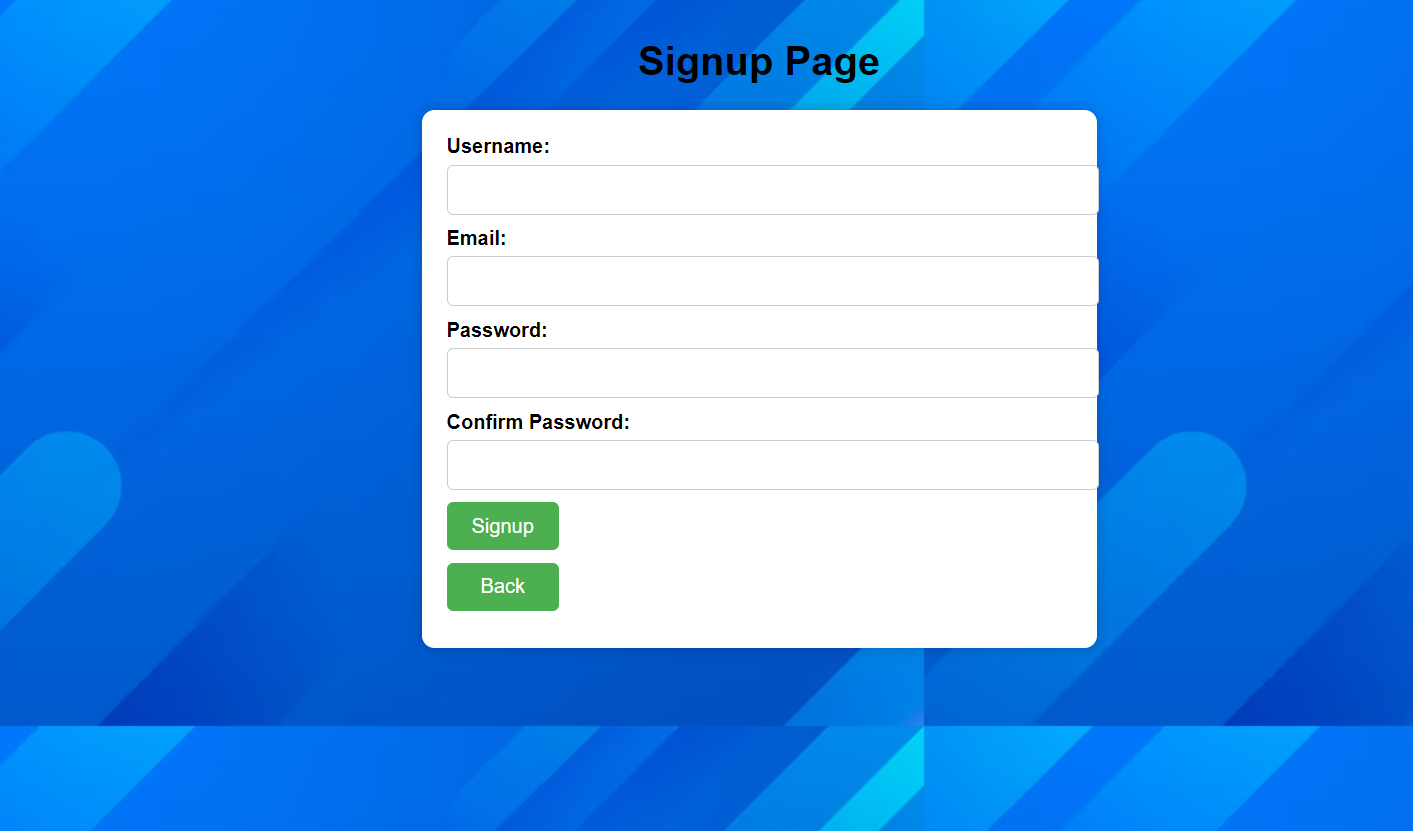


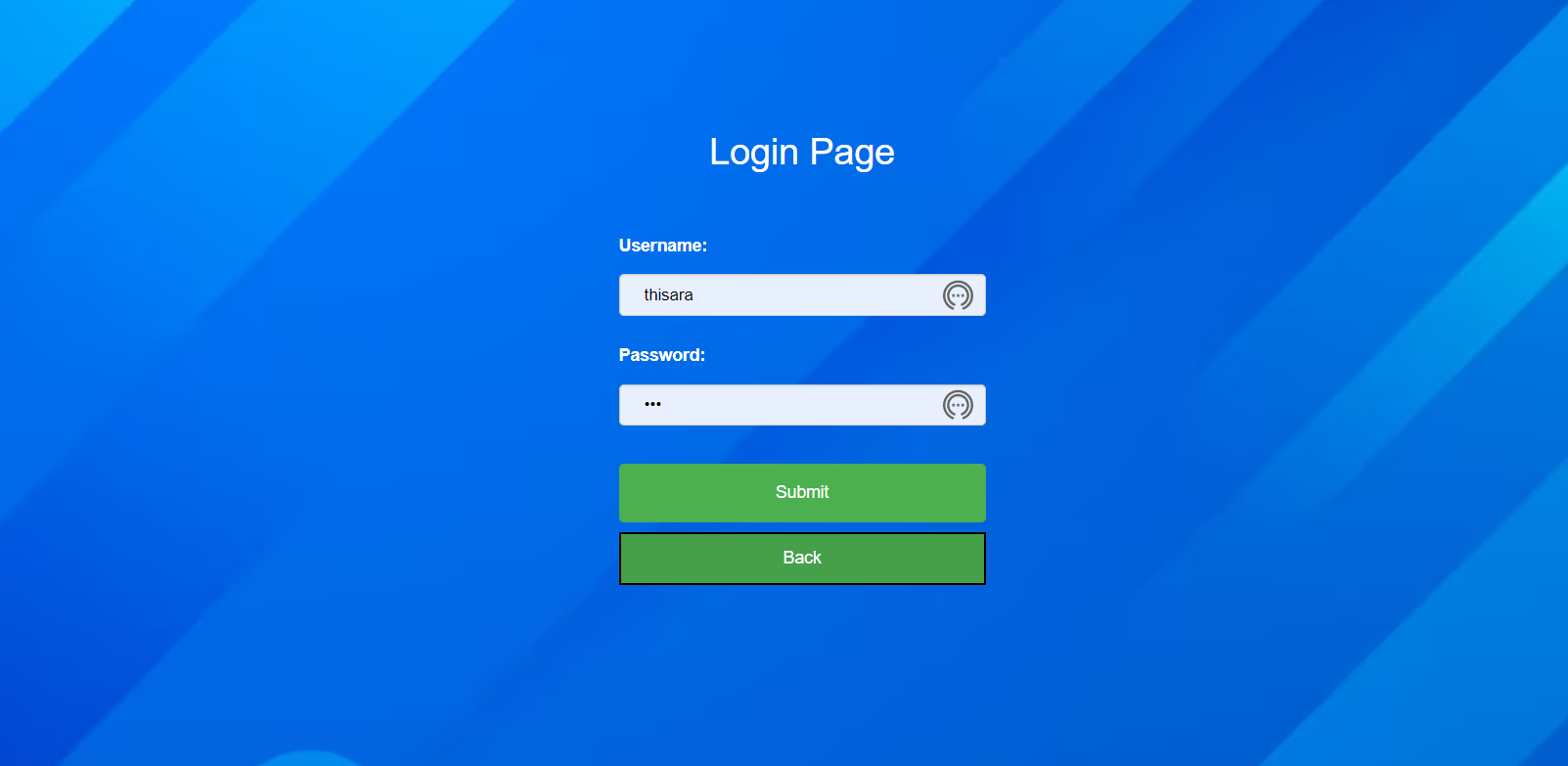
* **event.preventDefault();**: This prevents the default form submission behavior, ensuring that the form data is submitted via AJAX instead.
* **var username = $("#username").val();**: This retrieves the value of the username input field.
* **var email = $("#email").val();**: This retrieves the value of the email input field.
* **var password = $("#password").val();**: This retrieves the value of the password input field.
* **var confirmPassword = $("#confirm-password").val();**: This retrieves the value of the confirm password input

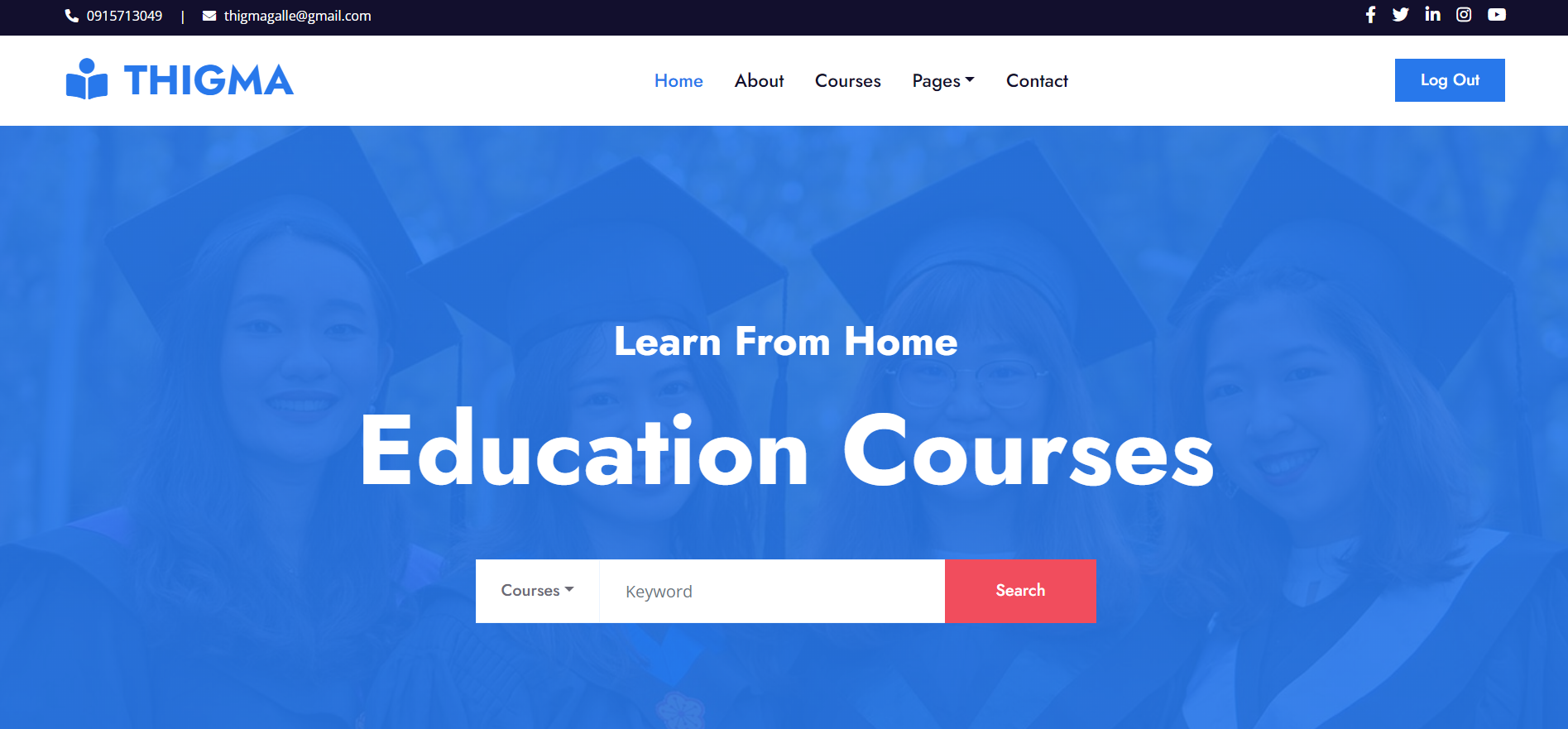


# **Screenshots of the website**



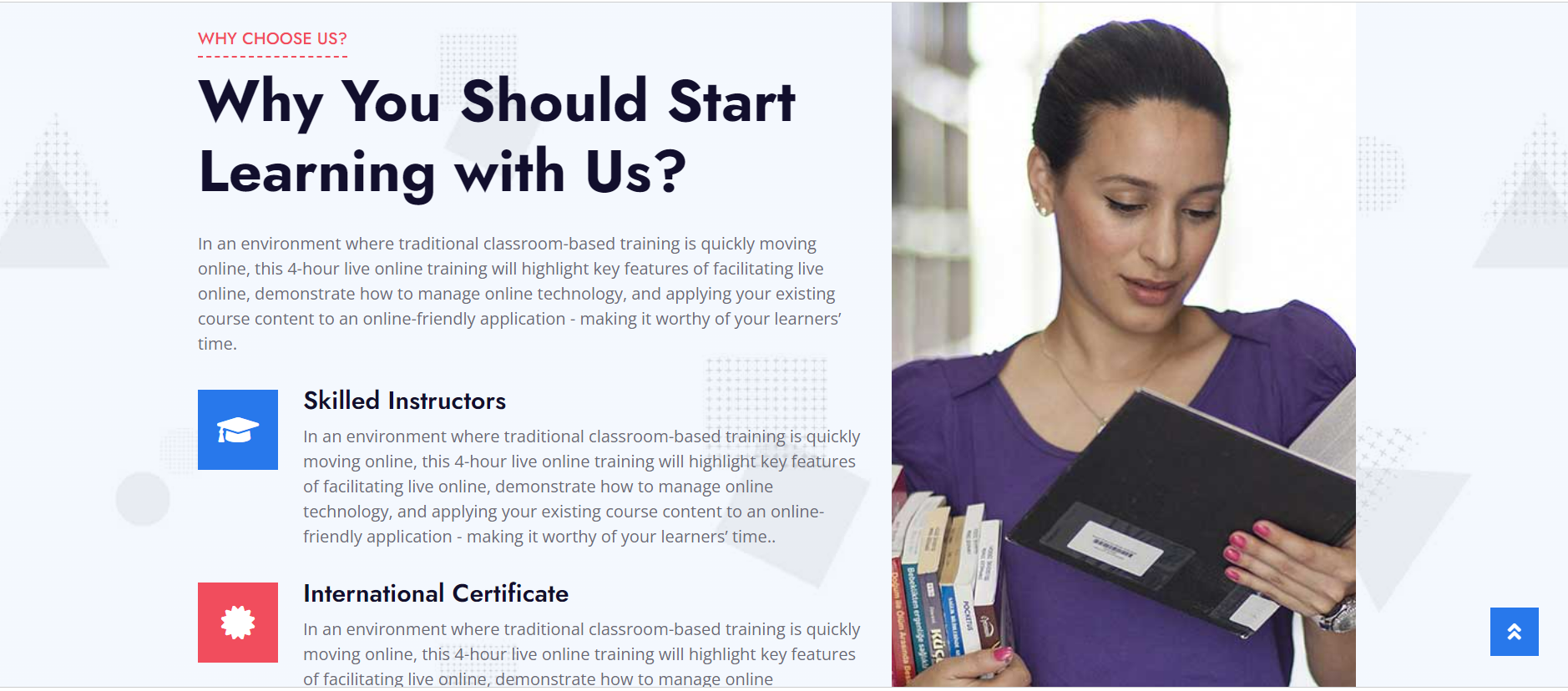






Website

Description automatically generated with medium confidence



Graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated