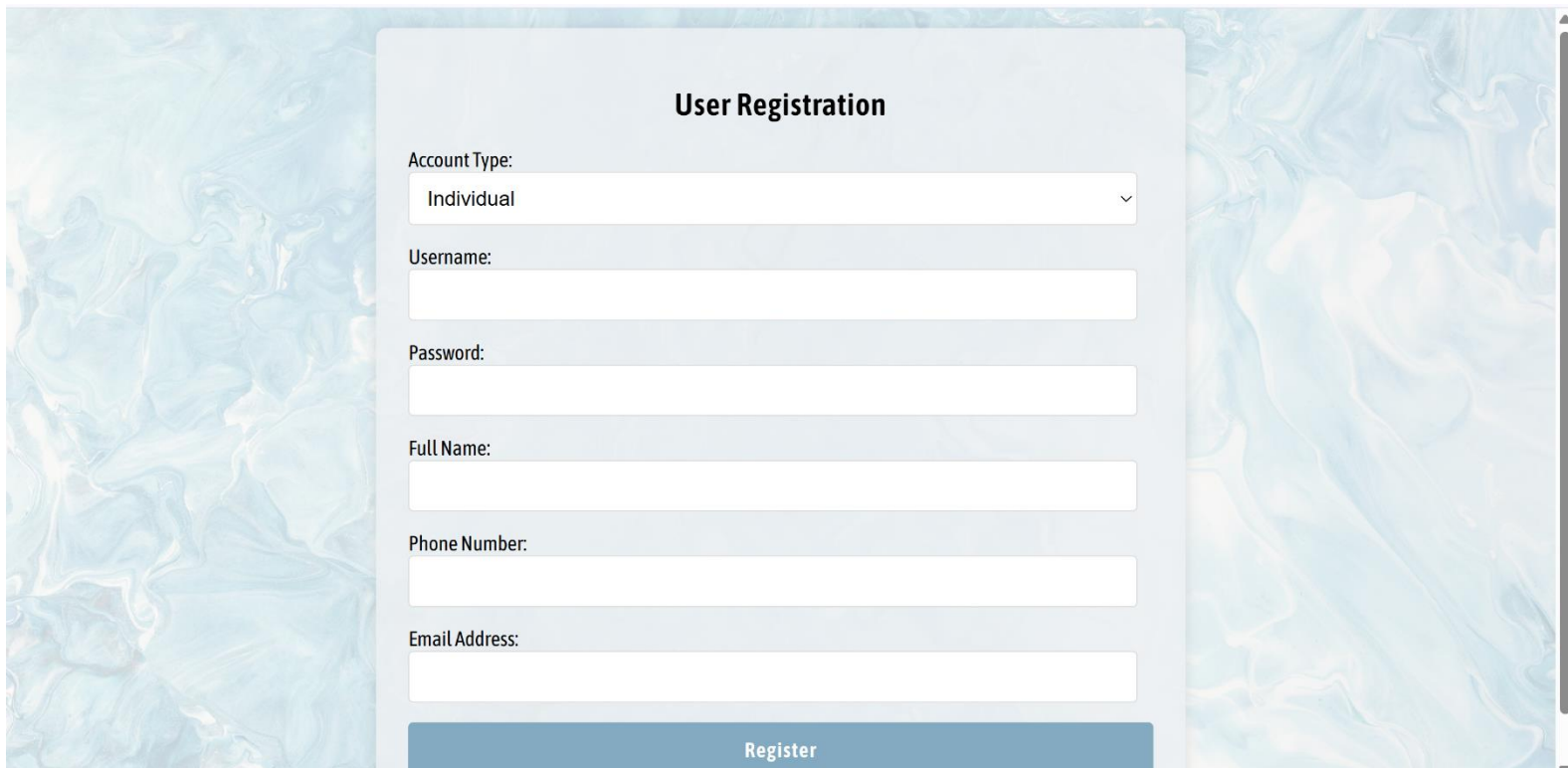


Secure User Registration System – Lorena Spallino

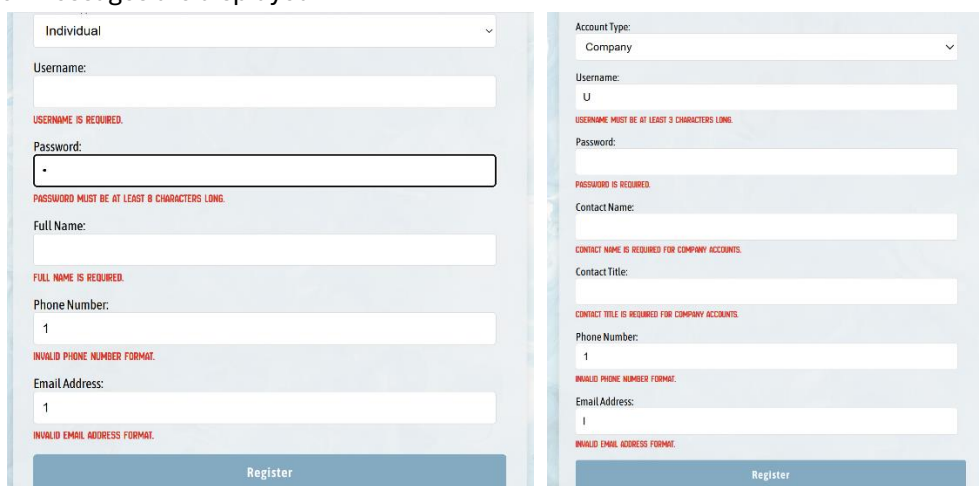
Using HTML, JavaScript, CSS, PHP and SQL, I have developed a visually appealing, secure, responsive web form that connects to a database, to store validated input. This report is intended to enhance the source code by displaying the output of the program, along with its features.

The first page, or landing page is made using HTML, CSS, and JavaScript. It displays a user-friendly visually appealing web form, with various inputs, a submit button and a select element for the user to choose between the individual account type and company. The background image has been manipulated with CSS to make it more transparent, for better visibility. Below is a screenshot of the form on the index.html page.



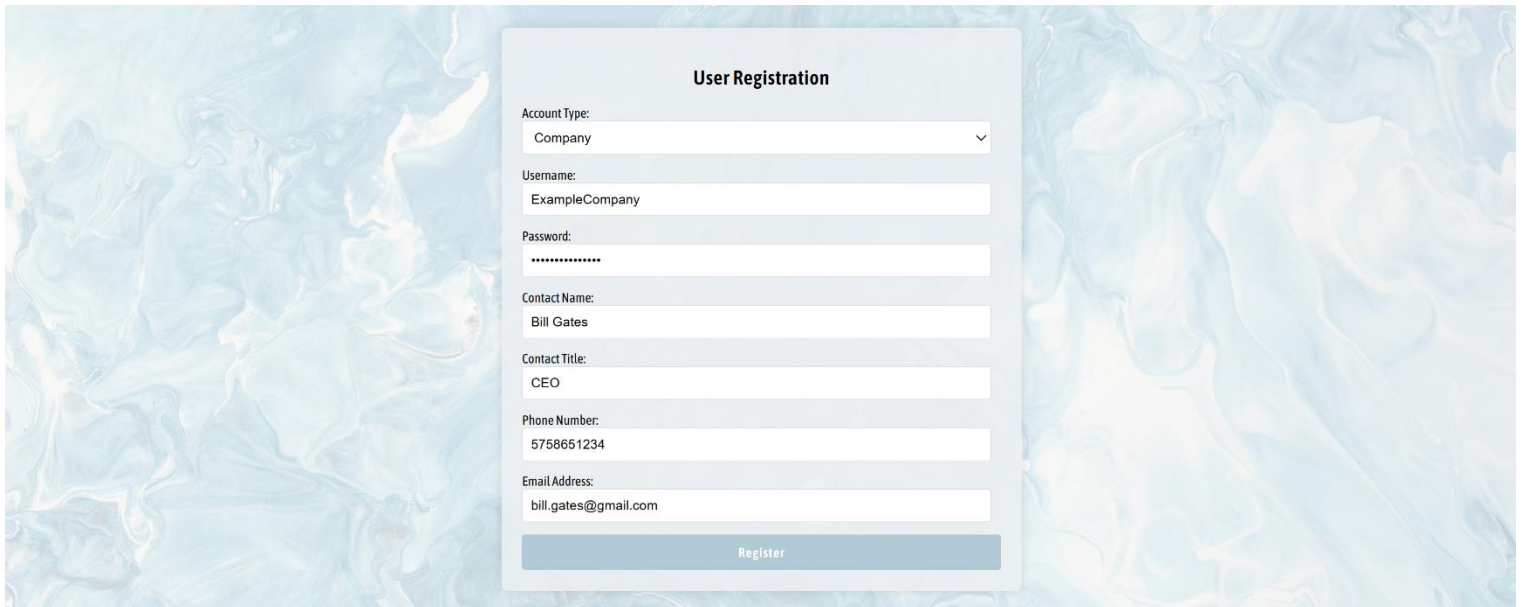
The screenshot shows a 'User Registration' form centered on a light blue marbled background. The form is a light gray box with a white border. It contains the following fields: 'Account Type' (a dropdown menu with 'Individual' selected), 'Username' (a text input), 'Password' (a text input), 'Full Name' (a text input), 'Phone Number' (a text input), and 'Email Address' (a text input). At the bottom of the form is a blue 'Register' button.

Using JavaScript functions, and event handlers, I have incorporated client-side validation, so that error messages are displayed if the user's inputs do not follow requirements. Below is a screenshot of how these error messages are displayed.



The image shows two side-by-side screenshots of the registration form with validation errors. The left screenshot shows the form with 'Individual' selected as the account type. The 'Username' field has the error 'USERNAME IS REQUIRED.' below it. The 'Password' field has the error 'PASSWORD MUST BE AT LEAST 8 CHARACTERS LONG.' below it. The 'Full Name' field has the error 'FULL NAME IS REQUIRED.' below it. The 'Phone Number' field has the error 'INVALID PHONE NUMBER FORMAT.' below it. The 'Email Address' field has the error 'INVALID EMAIL ADDRESS FORMAT.' below it. The right screenshot shows the form with 'Company' selected as the account type. The 'Username' field has the error 'USERNAME MUST BE AT LEAST 3 CHARACTERS LONG.' below it. The 'Contact Name' field has the error 'CONTACT NAME IS REQUIRED FOR COMPANY ACCOUNTS.' below it. The 'Contact Title' field has the error 'CONTACT TITLE IS REQUIRED FOR COMPANY ACCOUNTS.' below it. The 'Phone Number' field has the error 'INVALID PHONE NUMBER FORMAT.' below it. The 'Email Address' field has the error 'INVALID EMAIL ADDRESS FORMAT.' below it. Both screenshots have a blue 'Register' button at the bottom.

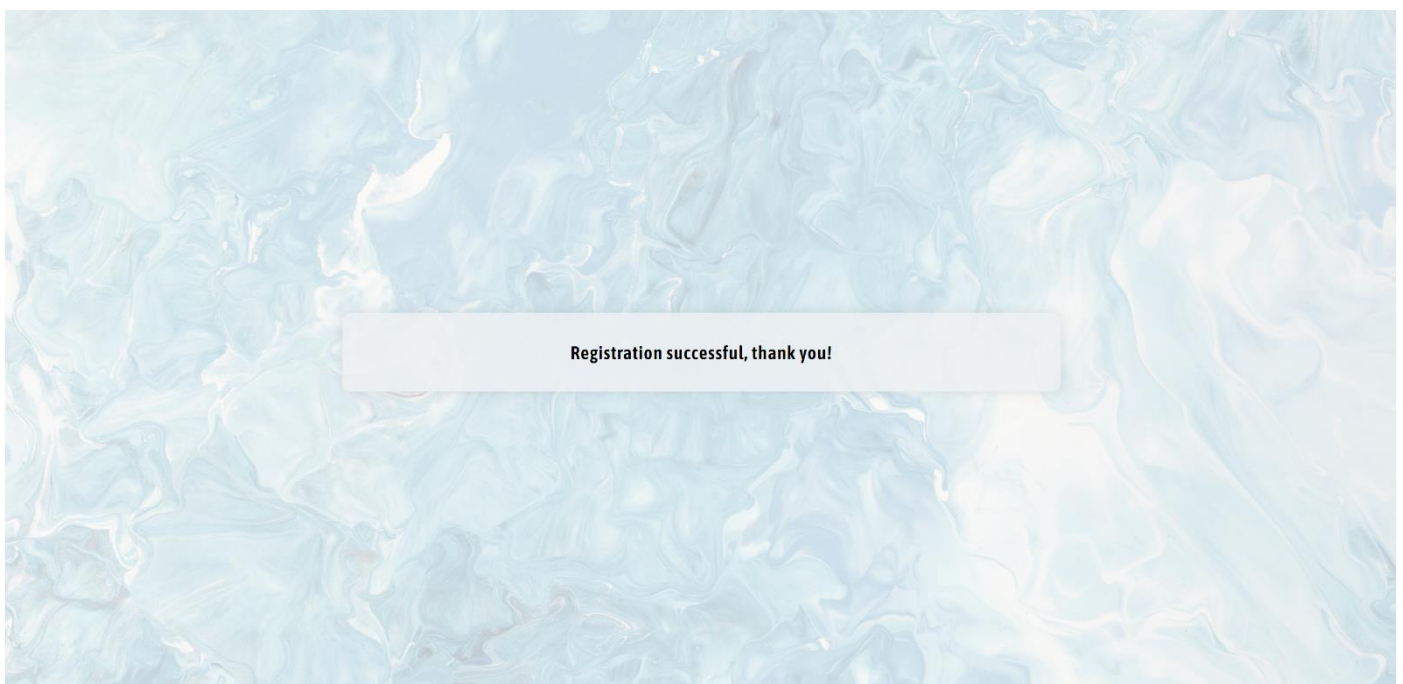
When the user selects the account type 'company', the form adapts to include the field 'contact name' instead of 'full name' and 'contact title'. This is accomplished through JavaScript functions, and styling. The screenshot below displays how the form adapts to the Company Account Type, and how the completed form looks when the input is valid.



A screenshot of a 'User Registration' form. The form is titled 'User Registration' and is set against a light blue background. It contains several input fields: 'Account Type' (a dropdown menu with 'Company' selected), 'Username' (text input with 'ExampleCompany'), 'Password' (password input with masked characters), 'Contact Name' (text input with 'Bill Gates'), 'Contact Title' (text input with 'CEO'), 'Phone Number' (text input with '5758651234'), and 'Email Address' (text input with 'bill.gates@gmail.com'). A 'Register' button is at the bottom.

User Registration	
Account Type:	Company
Username:	ExampleCompany
Password:	*****
Contact Name:	Bill Gates
Contact Title:	CEO
Phone Number:	5758651234
Email Address:	bill.gates@gmail.com
Register	

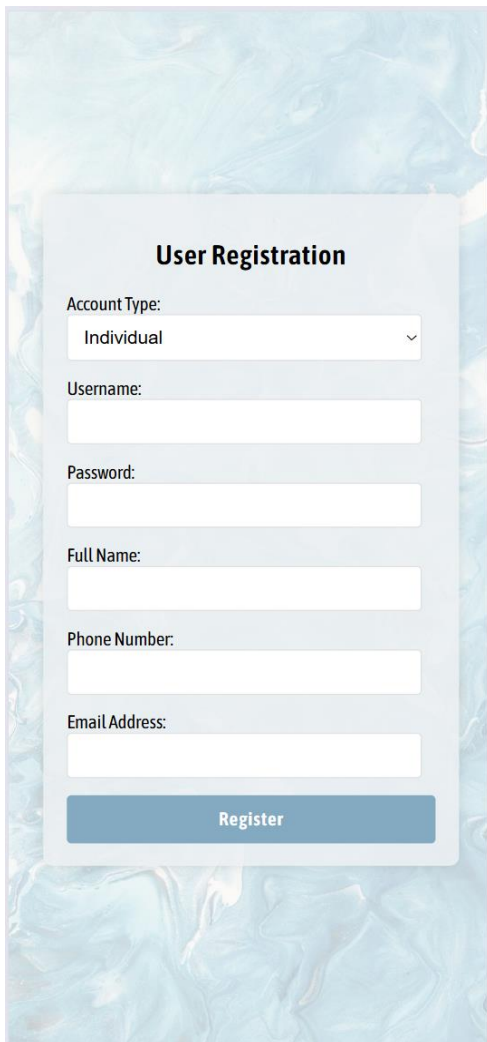
Upon submission, the form collects the data and sends it to the server using HTTP POST method. The web server receives the HTTP POST request. It identifies that the request is intended for the registration.php script, and then that script is executed. This script will again validate the inputs on the server-side to ensure that everything is valid. It sanitizes the data and hashes the password using php's built-in functions. It will then connect to the specified database, and using stored procedures and sql, it will update the user's registration information into the 'users' database. That script will then display a success message to the user, as shown in the screenshot below.



Finally, as shown in the following screenshot, upon successful submission of the form, and completion of the registration.php script, the database is updated with the validated information and the hashed password.

ID	account_type	username	password	full_name	phone_number	email_address	contact_name	contact_title
13	company	ExampleCompany	\$2y\$10\$.AekEYSHvsnkFfXN1aytuuhvdsIsVq31/96I2J/EMLw...		5758651234	bill.gates@gmail.com	Bill Gates	CEO

This web form has been made responsive using CSS media queries. The screenshots below demonstrate the look of the site on a mobile device.



User Registration

Account Type:
Individual ▾

Username:

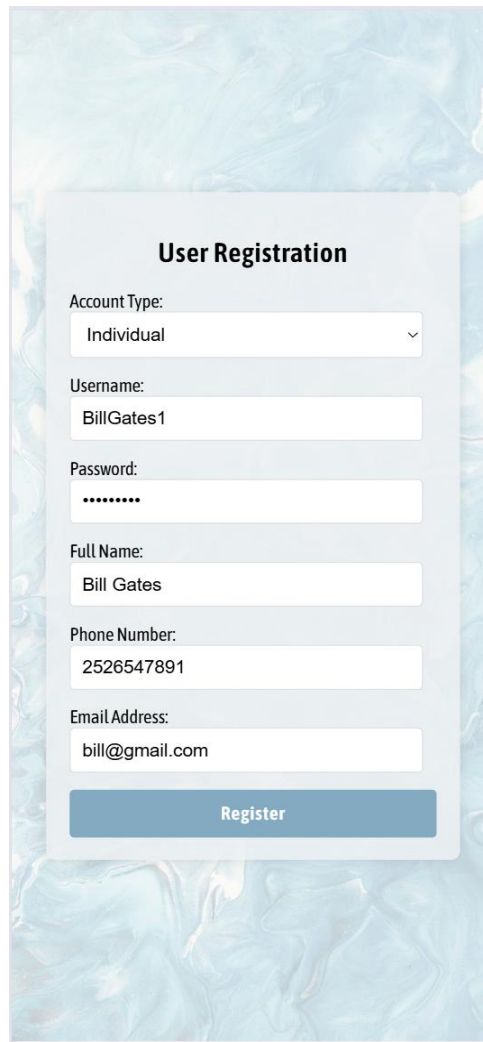
Password:

Full Name:

Phone Number:

Email Address:

Register



User Registration

Account Type:
Individual ▾

Username:
BillGates1

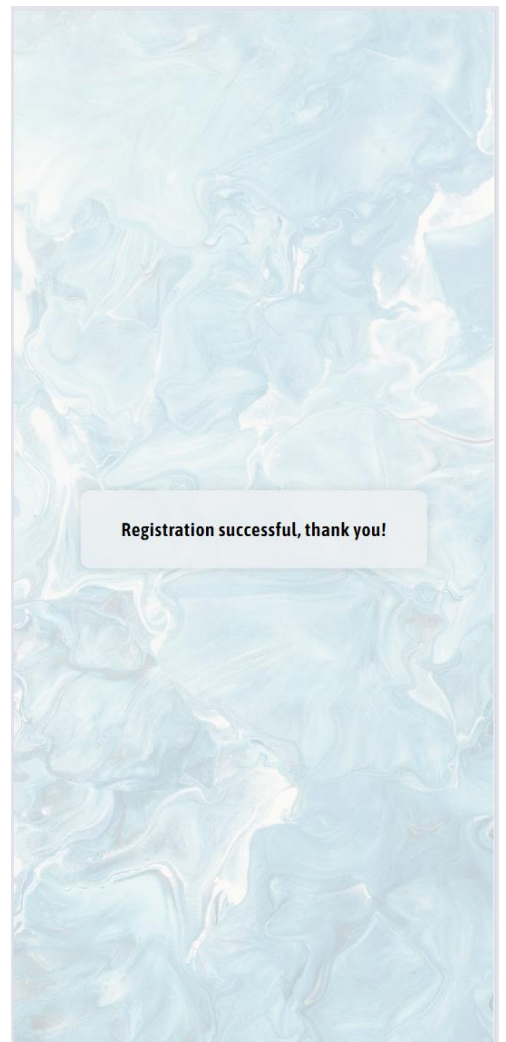
Password:

Full Name:
Bill Gates

Phone Number:
2526547891

Email Address:
bill@gmail.com

Register



Registration successful, thank you!