**Experiment 10**

**Title :** Form validation using PHP regular expressions

**Aim :** To make student understand the concept of regular expressions and how to use them in form validation

**Form Validation Using Regular Expressions in PHP**

PHP regular expressions (regex) are a powerful tool for pattern matching and string manipulation. PHP supports two types of regular expressions: POSIX and Perl-compatible regular expressions (PCRE). Most modern PHP applications use PCRE due to its flexibility and powerful features.

Key Concepts of PHP Regular Expressions

Basic Syntax:

* Regular expressions are enclosed in delimiters, usually slashes /.
* Example: /pattern/.

Metacharacters:

* .: Matches any character except a newline.
* ^: Asserts the start of a line.
* $: Asserts the end of a line.
* \*: Matches 0 or more occurrences of the preceding element.
* +: Matches 1 or more occurrences of the preceding element.
* ?: Matches 0 or 1 occurrence of the preceding element.
* {n}: Matches exactly n occurrences of the preceding element.
* {n,}: Matches n or more occurrences of the preceding element.
* {n,m}: Matches between n and m occurrences of the preceding element.

Character Classes:

* [abc]: Matches any one of the characters a, b, or c.
* [^abc]: Matches any character except a, b, or c.
* [a-z]: Matches any lowercase letter from a to z.
* [A-Z]: Matches any uppercase letter from A to Z.
* [0-9]: Matches any digit.

Escaping Characters:

* Use a backslash \ to escape metacharacters if you want to match them literally.
* Example: \. matches a literal dot.

Quantifiers:

* \*, +, ?, {n}, {n,}, and {n,m} specify how many times the preceding element should be matched.

Anchors:

* ^ (caret) asserts the position at the start of a string.
* $ (dollar) asserts the position at the end of a string.

Grouping and Alternation:

* Parentheses () are used for grouping.
* The pipe | acts as a logical OR.
* Example: (cat|dog) matches either "cat" or "dog".

**PHP Functions for Regular Expressions**

1. preg\_match(): Checks if a pattern matches a string. Returns 1 if there is a match, 0 if not, and FALSE if an error occurred.

if (preg\_match("/pattern/", $subject)) {

// Match found

}

1. preg\_match\_all(): Finds all matches of a pattern in a string. Returns the number of matches found.

preg\_match\_all("/pattern/", $subject, $matches);

1. preg\_replace(): Replaces matched patterns with a specified replacement string.

$result = preg\_replace("/pattern/", "replacement", $subject);

1. preg\_split(): Splits a string by a regular expression.

$result = preg\_split("/pattern/", $subject);

1. preg\_grep(): Filters an array by a regex pattern.

$result = preg\_grep("/pattern/", $array);

<?php

// Sample user input

$email = "user@example.com";

$username = "user123";

$password = "Passw0rd!";

$text = "John Doe, 29, john.doe@example.com; Jane Smith, 34, jane.smith@example.com";

// Function to validate email

function validateEmail($email) {

return preg\_match("/^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/", $email);

}

// Function to validate username

function validateUsername($username) {

return preg\_match("/^[a-zA-Z0-9]{5,15}$/", $username);

}

// Function to validate password

function validatePassword($password) {

return preg\_match("/^(?=.\*[A-Za-z])(?=.\*\d)[A-Za-z\d]{8,}$/", $password);

}

// Function to parse the text

function parseText($text) {

// Regex to match name, age, and email

preg\_match\_all("/([A-Za-z\s]+),\s\*(\d+),\s\*([a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,})/", $text, $matches);

return $matches;

}

// Validate user inputs

$isEmailValid = validateEmail($email);

$isUsernameValid = validateUsername($username);

$isPasswordValid = validatePassword($password);

// Output validation results

echo "Email validation: " . ($isEmailValid ? "Valid" : "Invalid") . PHP\_EOL;

echo "Username validation: " . ($isUsernameValid ? "Valid" : "Invalid") . PHP\_EOL;

echo "Password validation: " . ($isPasswordValid ? "Valid" : "Invalid") . PHP\_EOL;

// Parse the text

$parsedData = parseText($text);

if (!empty($parsedData[0])) {

echo "Parsed Information:\n";

foreach ($parsedData[0] as $index => $match) {

echo "Name: " . $parsedData[1][$index] . ", Age: " . $parsedData[2][$index] . ", Email: " . $parsedData[3][$index] . PHP\_EOL;

}

} else {

echo "No matches found." . PHP\_EOL;

}

?>

**Conclusion:-**

Thus, we have studied, understood and practically checked regular expression concepts in PHP