To prevent automatic user registrations and ensure that users are human, you can implement several Proof of Concept (POC) techniques. Here’s a general outline of how you can achieve this using CAPTCHA, along with some additional methods:

**1. Implementing CAPTCHA**

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a widely used method to prevent automated bots from registering. Here’s how to implement Google reCAPTCHA:

**a. Get reCAPTCHA keys:**

1. Go to the Google reCAPTCHA website.
2. Sign in with your Google account.
3. Register your site to get the site key and secret key.

**b. Add reCAPTCHA to your registration form:**

Include the reCAPTCHA widget in your HTML form.

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<title>Register</title>

<script src="https://www.google.com/recaptcha/api.js" async defer></script>

</head>

<body>

<form action="submit\_registration.php" method="post">

<label for="username">Username:</label>

<input type="text" id="username" name="username"><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email"><br>

<label for="password">Password:</label>

<input type="password" id="password" name="password"><br>

<div class="g-recaptcha" data-sitekey="YOUR\_SITE\_KEY"></div><br>

<input type="submit" value="Register">

</form>

</body>

</html>

**c. Validate reCAPTCHA in the backend:**

Verify the CAPTCHA response on your server-side script.

php

Copy code

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$recaptcha\_secret = 'YOUR\_SECRET\_KEY';

$recaptcha\_response = $\_POST['g-recaptcha-response'];

// Make a request to the reCAPTCHA server

$response = file\_get\_contents("https://www.google.com/recaptcha/api/siteverify?secret=$recaptcha\_secret&response=$recaptcha\_response");

$responseKeys = json\_decode($response, true);

if(intval($responseKeys["success"]) !== 1) {

echo "Please complete the CAPTCHA.";

} else {

// Process registration

echo "Registration successful!";

}

}

?>

**2. Additional Techniques**

**a. Honeypot Fields:**

Include a hidden field in your form that humans won't fill out but bots might.

html

Copy code

<input type="text" name="website" id="website" style="display:none">

Check on the server side to ensure the field is empty.

php

Copy code

if (!empty($\_POST['website'])) {

die("Bot detected!");

}

**b. Time-Based Form Submission:**

Measure the time taken to fill out the form. Bots usually fill out forms instantly.

php

Copy code

// Start time when form is loaded

$startTime = time();

// Store start time in session

$\_SESSION['start\_time'] = $startTime;

Check the submission time on the server.

php

Copy code

$startTime = $\_SESSION['start\_time'];

$endTime = time();

$timeTaken = $endTime - $startTime;

if ($timeTaken < 5) { // Adjust time threshold as needed

die("Form submitted too quickly. Possible bot.");

}

**c. Email Verification:**

Require users to verify their email address by clicking a link sent to their inbox.

1. Generate a unique verification token.
2. Send an email with the token link.
3. Activate the account only after the link is clicked.

**Conclusion**

By combining CAPTCHA with other techniques like honeypot fields, time-based submission checks, and email verification, you can significantly reduce the risk of automated registrations. Implementing these measures in your registration process will help ensure that your users are real humans.