**reCAPTCHA Usage Guide (EN)**

**Purpose:**

**This document explains how to integrate the reCAPTCHA component into the login page and other pages, as well as how to configure and reuse it across different parts of the application.**

**1. General Usage Steps**

**The GoogleReCaptchaProvider component is active only when RECAPTCHA\_CONFIG.ENABLED is set to true. The component supports both reCAPTCHA v3 and v2 in a hybrid way.**

**Key Updates:**

1. **validateV3Token Integration: The validateV3Token service is used to validate the v3 token via Axios, which communicates with the backend.**
2. **Score Check and v2 Rendering: If the returned score is low (e.g., score < threshold), the v2 component is automatically rendered.**

**2. validateV3Token Usage**

**The validateV3Token function checks the validity of the v3 token sent to the backend and returns the validation result, including the reCAPTCHA score.**

**Example Service Function:**

**File Path: src/services/captcha/index.api.ts**

**import { AxiosInstance } from 'axios';**

**import { RootResponse, ValidateResponse } from '../../utils/types';**

**export const validateV3Token = async (data: { token: string }, http: AxiosInstance) => {**

**const response = await http.post<RootResponse<ValidateResponse>>('/validate-v3-token', data);**

**return response.data;**

**};**

**Response Type:**

* **success: Indicates whether the validation was successful.**
* **result.score: The reCAPTCHA score (between 0 and 1).**
* **action: The action name for reCAPTCHA v3 (optional).**

**3. Login Page Usage**

**The login component is configured as follows. Only essential code snippets are shown.**

**Key Code Snippets:**

**Login.tsx:**

**const onSubmit = (data: LoginFormInputs) => {**

**if (RECAPTCHA\_CONFIG.ENABLED) {**

**if (v3Token) {**

**loginMutation.mutate({ ...data, http, recaptchaV3Token: v3Token });**

**} else {**

**const v2Token = recaptchaRef.current?.getValue();**

**if (!v2Token) {**

**showNotification({**

**severity: "error",**

**summary: "Error",**

**detail: "Captcha Required",**

**});**

**return;**

**}**

**loginMutation.mutate({ ...data, http, recaptchaV2Token: v2Token });**

**}**

**} else {**

**loginMutation.mutate({ ...data, http });**

**}**

**recaptchaRef.current?.reset();**

**};**

**Form Render:**

**<GoogleReCaptchaProvider reCaptchaKey={RECAPTCHA\_CONFIG.SITE\_KEY\_V3}>**

**<form onSubmit={handleSubmit(onSubmit)}>**

**<InputText {...register("email", { required: true })} />**

**<InputText type="password" {...register("password", { required: true })} />**

**{RECAPTCHA\_CONFIG.ENABLED && RECAPTCHA\_CONFIG.SITE\_KEY\_V2 && (**

**<Captcha**

**siteKey={RECAPTCHA\_CONFIG.SITE\_KEY\_V2}**

**ref={recaptchaRef}**

**action="login\_action"**

**onChange={(token) => setV3Token(token)}**

**threshold={0.5}**

**http={http}**

**/>**

**)}**

**<Button label="Login" />**

**</form>**

**</GoogleReCaptchaProvider>**

**4. RECAPTCHA\_CONFIG Configuration**

**File Path: src/configs/captcha.tsx**

**export enum RecaptchaSize {**

**NORMAL = "normal",**

**COMPACT = "compact",**

**INVISIBLE = "invisible",**

**}**

**export const RECAPTCHA\_CONFIG = {**

**ENABLED: true,**

**SITE\_KEY\_V3: import.meta.env.VITE\_SITE\_KEY\_V3 || "",**

**SITE\_KEY\_V2: import.meta.env.VITE\_SITE\_KEY\_V2 || "",**

**SIZE: RecaptchaSize.NORMAL,**

**};**

**5. Working Without reCAPTCHA**

* **You can disable reCAPTCHA by setting ENABLED to false.**
* **For example, in the development environment, you can leave the SITE\_KEY empty in your .env file:**

**VITE\_SITE\_KEY\_V3=**

**VITE\_SITE\_KEY\_V2=**

* **In this case, the form will be rendered without reCAPTCHA verification.**

**6. Summary**

**This guide explains how to use the reCAPTCHA component in the login page. The Captcha component validates tokens via Axios and dynamically switches between v2 and v3 based on the reCAPTCHA score. You can reuse the Captcha component on other pages for secure verification as well.**