go_api

Golang API

function	endpoint	argument	result	detail
frontend				

ユーザーを保存するAPI	/users/create	{
		user_id: string,
		mail_address: string,
		password: string,
		}
			・Success case
			{
			"StatusMessage": "Success”,
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Failed”
				1, golang側に与えられたデータの形式がJSONでなかった
				message: “JSON形式ではありません”
				2, メールアドレスが既に登録されている場合
				message: “このメールアドレスは既に登録されています”
				3, データベースにエラーが発生した場合
				message: "データベースエラーが発生しました”
				4, データベースに保存できなかった場合
				message: "データベースに保存できませんでした”

ユーザーを認証するAPI	/users/login	{
		mail_address: string,
		password: string,
		}
			・Success case
			{
			"StatusMessage": "Success”,
			“UserId”: string
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Failed”
				1, メールアドレスがデータベースに存在しない場合
				message: “このメールアドレスは存在しません”
				2, パスワードが違う場合
				message: “パスワードが違います！”

ユーザーを削除するAPI	/users/delete	{
		user_id: string
		}
			・Success case
			{
			"StatusMessage": "Success”
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Failed”
				1, メールアドレスがデータベースに存在しない場合
				message: “このメールアドレスは存在しません”
				2, データベースにuser_idが存在しなかった場合
				message: “Useridが存在しません”
				3, ユーザーを削除できなかった場合
				message: “ユーザーを削除できませんでした”

Passwordを変更するAPI	/users/change/password	{
		user_id: string,
		password: string,
		new_password: string,
		}
			・Success case
			{
			"StatusMessage": "Success”,
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Failed”
				1, golang側に与えられたデータの形式がJSONでなかった
				message: “JSON形式ではありません”
				2, データベースにuser_idが存在しなかった
				message: “Useridが存在しません”
				3, 更新前のパスワードが違った
				message: "パスワードが違います！”
				4, パスワードを更新できなかった
				message: “パスワードを更新できませんでした”

GeminiApiKeyを保存するAPI	/users/save/api	{
		user_id: string,
		gemini_api_key: string,
		}
			・Success case
			{
			"StatusMessage": "Success”,
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラ
				・StatusMessage == “Failed”
				1, golang側に与えられたデータの形式がJSONでなかった
				message: “JSON形式ではありません”
				2, データベースにuser_idが存在しなかった
				message: “Useridが

			ーメッセージ", "error": “デバッグ用エラーメッセージ”, }	存在しません” 3, GeminiApiKeyを保存できなかった message: "GeminiApiKeyを保存できませんでした”

Passwordを認証するAPI	/users/check/password	{
		user_id: string,
		password: string
		}
			・Success case
			{
			"StatusMessage": "Success”,
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Failed”
				1, golang側に与えられたデータの形式がJSONでなかった
				message: “JSON形式ではありません”
				2, データベースにuser_idが存在しなかった
				message: “Useridが存在しません”
				3, passwordが一致しなかった
				message: "passwordが一致しません”

GeminiApiKeyを取得するAPI	/users/get/api	{
		user_id: string
		}
			・Success case
			{
			"StatusMessage": "Success”,
			“GeminiApiKey”: string
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Failed”
				1, golang側に与えられたデータの形式がJSONでなかった
				message: “JSON形式ではありません”
				2, データベースにuser_idが存在しなかった
				message: “Useridが存在しません”

データベース内のユーザー情報を確認するAPI	/users/check/user	{
		mail_address: string
		}
			・Success case
			{
			"StatusMessage": "Success”,
			“case”: string
			}
			
			・Failed case
			{
				・StatusMessage == “Success”
				1, mail_addressがデータベース内に存在しなかった。
				case: “possible”
				2, mail_addressがデータベースに存在した。

			"StatusMessage": "Failed", "massage": "エラーメッセージ", "error": “デバッグ用エラーメッセージ”, }	case: “impossible” 3, 復元可能な場合 case: “restoration” ・StatusMessage == “Failed” 1, golang側に与えられたデータの形式がJSONでなかった message: “JSON形式ではありません”

過去のユーザー情報を消し、新たにユーザーを登録するAPI	/users/recreate	{
		user_id: string,
		mail_address: string,
		password: string,
		}
			・Success case
			{
			"StatusMessage": "Success”,
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Failed”
				1, golang側に与えられたデータの形式がJSONでなかった
				message: “JSON形式ではありません”
				2, ユーザー情報を削除できなかった場合
				message: “ユーザー情報を削除できませんでした”
				3, データベースに保存できなかった場合
				message: "データベースに保存できませんでした”

過去のユーザー情報を復元するAPI	/users/restoration	{
		mail_address: string,
		password: string
		}
			・Success case
			{
			"StatusMessage": "Success”,
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Failed”
				1, golang側に与えられたデータの形式がJSONでなかった
				message: “JSON形式ではありません”
				2, mail_addressが存在しなかった場合
				message: “このメールアドレスは存在しません”
				3, passwordが一致しなかった
				message: "passwordが一致しません”
				4, ユーザー情報を復元出来なかった場合
				message: “ユーザー情報を復元できませんでした”

CSVファイルの情報を取得するAPI	/csvs/get	{
		user_id: string
		}
			・Success case
			{
			"StatusMessage": "Success”,
			“CsvData”: list,
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Success”
				CsvData: [
				{
				csv_id: string,
				file_name: string,
				data_size: int,
				data_columns: int,
				data_rows: int,
				last_accessed_date: string
				}
				]
				
				・StatusMessage == “Failed”
				1, golang側に与えられたデータの形式がJSONでなかった
				message: “JSON形式ではありません”
				2, データベースからデータが取得されなかった場合
				message: “データが取得されませんでした”

CSVファイルを削除するAPI	/csvs/delete	{
		csv_id: string
		}
			・Success case
			{
			"StatusMessage": "Success”,
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}
				・StatusMessage == “Failed”
				1, golang側に与えられたデータの形式がJSONでなかった
				message: “JSON形式ではありません”
				2, データベースからデータが取得されなかった場合
				message: “データが取得されませんでした”
				3, データベースに保存できなかった場合
				message: "データベースに保存できませんでした”

CSVファイルを復元するAPI	/csvs/restoration	{
		csv_id: string
		}
			・Success case
			{
			"StatusMessage": "Success”,
			}	

			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}	

backend				

csvファイルを取得するAPI	/csvs/get/data:csv_id		・Success case
			{
			"StatusMessage": "Success”,
			“file_name”: string,
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}	

csvファイルを更新するAPI	/csvs/update	files = {
		csv_file: csv,
		json_file: json,
		}
		json = {
		csv_id: str,
		data_size: int,
		data_columns: int,
		data_rows: int,
		}
			・Success case
			{
			"file": {
			“csv_file”: csv,
			“json_file”: json
			}
			}
			
			・Failed case
			{
			"StatusMessage": "Failed",
			"massage": "エラーメッセージ",
			"error": “デバッグ用エラーメッセージ”,
			}	
