**RandomPasswordGenerator Composer Package**

The **RandomPasswordGenerator** package provides a simple and flexible solution for generating secure and customizable random passwords. This package is designed to be easily integrated into your Laravel projects, allowing you to create strong passwords according to specified rules.

\*GitHub project link: https://github.com/toystorya705/intellicore\_test\_acme\_project

**Installation**

* Add package to **RandomPasswordGenerator** file to your project.
* Add the following to your projects’ composer.js
* "autoload": {
* "psr-4": {
* "RandomPasswordGenerator\\": "random-password-generator/src/",
* "CodeService\\": "random-password-generator/src"
* }
* },
* To install the package, use Composer: Composer install
* Then add migration access\_code migration file to your migration folder.
* Then run: php artisan migrate
* Then setup AccessCode Model

\* File location in the project

Migration file: \database\migrations\2023\_08\_14\_030013\_create\_access\_codes\_table.php

Model file: \app\Models\AccessCode.php

**Usage**

The package contains two classes RandomPasswordGenrator and CodeService

\*Package is location: \random-password-generator

**RandomPasswordGenrator**

**1. generate($length)**

This function generates a random password of the specified length using a set of characters (e.g., alphanumeric) and returns the generated password as a string. The function takes one parameter:

$length (int): The length of the password to generate.

**2. generateWithCondition($length, $userId)**

This function generates a random password that meets certain conditions, such as avoiding palindromes, repeated characters, sequences, and ensuring a minimum number of unique characters. It iteratively generates passwords until a valid one is found and returns it. The function takes two parameters:

$length (int): The length of the password to generate.

$userId (int): An identifier for context-specific conditions (optional).

**3. generateRandomPasswordWithCheckInDatabase()**

This function generates a random password that is not already in the database. It iteratively generates passwords until a unique one is found and returns it. This is useful for generating unique access codes. No parameters are required for this function.

**4. isPalindrome($string)**

This function checks if a given string is a palindrome, regardless of case and spaces. It returns a boolean value:

$string (string): The string to check for palindrome.

**5. hasRepeatedCharacters($string, $maxRepeats)**

This function checks if a string has any character repeated more than the specified limit. It returns a boolean value:

$string (string): The string to check for repeated characters.

$maxRepeats (int): The maximum number of allowed repeated occurrences for each character.

**6. hasSequentialCharacters($code, $maxSequenceLength)**

This function checks if a string has sequential characters (e.g., "123", "456") longer than the specified limit. It returns a boolean value:

$code (string): The string to check for sequential characters.

$maxSequenceLength (int): The maximum allowed length of sequential characters.

**7. countUniqueCharacters($string)**

This function counts the number of unique characters in a given string and returns the count:

$string (string): The string to count unique characters in.

**CodeService**

CodeService package, which is designed for managing access codes for users. Here's an overview of each function provided by the package:

**1. allocateCodeToUser($userId)**

This function allocates a new access code to a user if one hasn't already been allocated. It ensures that a user doesn't have more than one code allocated. The function returns the newly allocated code as a string, or null if the user already has an allocated code.

$userId (int): The identifier of the user to allocate a code to.

**2. resetAllocatedCodeToUser($userId)**

This function resets an allocated code for a user, generating a new code for them. It returns the new code as a string, or null if the user is not found.

$userId (int): The identifier of the user to reset the allocated code for.

**3. checkCode($code)**

This function checks if a given code exists in the system, returning a boolean value. It's useful for validating whether a code is valid and active.

$code (string): The access code to check.

**4. getAllocateCodeToUser($userId)**

This function retrieves the allocated code for a specific user. If no code is allocated to the user, it returns null.

$userId (int): The identifier of the user to retrieve the allocated code for.

**5. deleteAllocateCodeToUser($userId)**

This function deletes the allocated code for a user, essentially resetting their access code. It returns a success message as a string if the code is deleted, or null if the user is not found.

$userId (int): The identifier of the user to delete the allocated code for.

In order to import the classes, use the following

use RandomPasswordGenerator\RandomPasswordGenerator;

use CodeService\CodeService;

rt the classes

**Testing**

* The **generate** function generates passwords of the correct length.
* The **isPalindrome** function correctly identifies palindromes.
* The **hasRepeatedCharacters** function detects repeated characters as expected.
* The **hasSequentialCharacters** function detects sequential characters accurately.
* The **countUniqueCharacters** function calculates the correct count of unique characters.
* The **allocateCodeToUser** function allocates a code correctly and handles existing allocations.
* The **resetAllocatedCodeToUser** function resets a user's allocated code and generates a new one.
* The **checkCode** function correctly identifies the existence of a code.
* The **getAllocateCodeToUser** function retrieves the allocated code for a user accurately.
* The **deleteAllocateCodeToUser** function deletes a user's allocated code and handles non-existent users.

To run the test :

php artisan test

or

vendor/bin/phpunit --testdox tests

**Apart From Project Requirement**

An API has been set up

<https://intellicoretestacmeproject-production.up.railway.app/generate-random-password>

<https://intellicoretestacmeproject-production.up.railway.app/generate-random-password/checkUser>

<https://intellicoretestacmeproject-production.up.railway.app/generate-random-password/resetCode>

<https://intellicoretestacmeproject-production.up.railway.app/generate-random-password/deleteUser>

Use Json below in the body while making request

{

    "user\_id":14

}

**------------------------------------------------------------------------------------------------------------**

**Check code for access**

<https://intellicoretestacmeproject-production.up.railway.app/generate-random-password/checkCode>

{

    "code":807538

}

You can verify the code.