

ChatGPT

Explore GPTs

Today

Get Job ID from URL

Yesterday

Dynamic API Update

Efficient Order Search Enhanc ...

Date Format Explanation

Previous 7 Days

Resignation Letter Assistance

Resignation Letter Advice

Resignation from Akbar Tech

Phase 2 Style Update

Update Style Amount Calculation

Validation Code Based on Migrat

Proxy Configuration Rewrite Con

Default Value for Query

Generate 100 Orders Seeder

Busy Apologies Sent

Handling Undefined Category Re

Template Complexity Update

I want the order search to be more precise and efficient, here is my normal code snippet,

```
public function search()
{
    $search = request('search');

    $order = Order::paginate(20);

    return response()->json([
        'data' => $order
    ]);
}
```

Conditions:

1. It's a get request
- 2.

Like this `$search = request('search');`, this is an example , the user can search by,

1. email
2. order_status
3. payment_status
4. editor
5. start_date
6. end_date
7. paginate

Now the condition

1. Suppose the user can search with only one or two parameters like email, order_status, the inputs should be dynamic
2. The search will be efficient, for example email is tamhid.tff@gmail.com, the search query should be like this %amh%,
3. the default paginate value is 10, if query parameter gets the value like

```
$paginate = request('paginate');
```

Then the value of paginate will be changed

4. There is two parameters which is, start_date and end_date, if there is no query as end date, it will also consider end_date as start_date
5. The response data contains some of the data like,

```
"category_id": 2,
"editors_id": "1",
"styles_array": "[13,14]",
```

For the category_id, find it from Category model, also if its trashed with soft deletes, it will show the value with `trashed()`

Same for editors_id, find it from Editor model
Same for styles_array, Find it from Style model

To make your search functionality more precise and efficient, you can update your code to dynamically handle different search parameters, implement partial matching for email, set default pagination, handle date ranges, and fetch related data from other models. Below is the improved code snippet:

```
php
public function search()
{
    $query = Order::query();

    $searchParams = [
        'email' => request('email'),
        'order_status' => request('order_status'),
        'payment_status' => request('payment_status'),
        'editor' => request('editor'),
        'start_date' => request('start_date'),
        'end_date' => request('end_date'),
    ];

    if ($searchParams['email']) {
        $query->where('email', 'LIKE', '%' . substr($searchParams['email'], 0, -1) . '%');
    }

    if ($searchParams['order_status']) {
        $query->where('order_status', $searchParams['order_status']);
    }

    if ($searchParams['payment_status']) {
        $query->where('payment_status', $searchParams['payment_status']);
    }

    if ($searchParams['editor']) {
        $query->where('editor', $searchParams['editor']);
    }

    if ($searchParams['start_date']) {
        $end_date = $searchParams['end_date'] ?: $searchParams['start_date'];
        $query->whereBetween('created_at', [$searchParams['start_date'], $end_date]);
    }

    $paginate = request('paginate', 10);
    $orders = $query->paginate($paginate);
```

```

    $orders->getCollection()->transform(function ($order) {
        $order->category = Category::withTrashed()->find($order->category_id);
        $order->editor = Editor::find($order->editors_id);
        $order->styles = Style::whereIn('id', json_decode($order->styles_array))->get();
        return $order;
    });

    return response()->json([
        'data' => $orders
    ]);
}

```

Explanation:

1. Dynamic Search Parameters:

- The search parameters are stored in an array and checked for their existence.
- For the `email`, a `LIKE` query is used with the `%` wildcard for partial matching.

2. Pagination:

- The default pagination value is set to `10`. It can be overridden by passing the `paginate` parameter in the request.

[Get citation](#)

3. Date Range:

- If only `start_date` is provided, `end_date` defaults to the `start_date` to create a single-day range.

4. Related Data:

- The `category_id`, `editors_id`, and `styles_array` fields are used to fetch related data from the respective models.
- Soft deleted records are included for the `Category` model using `withTrashed()`.
- The `styles_array` field is expected to be a JSON encoded array of IDs.

This code ensures that your search functionality is efficient, handles various search parameters dynamically, and fetches related data appropriately.

this two also should be withTrashed output,

```

$order->editor = Editor::find($order->editors_id);
$order->styles = Style::whereIn('id', json_decode($order->styles_array))->get();

```

Just update the snippet and show it here, just the snippet



Here's the updated snippet with `withTrashed()` for the `Editor` and `Style` models:

```

php
↓
Copy code
$order->category = Category::withTrashed()->find($order->category_id);
$order->editor = Editor::withTrashed()->find($order->editors_id);
$order->styles = Style::withTrashed()->whereIn('id', json_decode($order->styles_array))->get();

```

[Get citation](#)

Please update this structure,

```

if ($searchParams['editor']) {
    $query->where('editor', $searchParams['editor']);
}

```

This should be searched by editors_name, since the value is in editor_id, map it first, find it from Editor model and then search it



Here's the updated code snippet to search by `editors_name` using the `editor_id`:

```

php
↓
Copy code
if ($searchParams['editor']) {
    $editor = Editor::where('name', 'LIKE', '%' . $searchParams['editor'] . '%')->first();
    if ($editor) {
        $query->where('editor', $editor->id);
    } else {
        // If no editor is found, set the query to return no results
        $query->whereNull('editor');
    }
}

```

[Get citation](#)

And the complete updated function with all the changes:

```

php
↓
Copy code
public function search()
{
    $query = Order::query();

    $searchParams = [
        'email' => request('email'),
        'order_status' => request('order_status'),
        'payment_status' => request('payment_status'),
        'editor' => request('editor'),
        'start_date' => request('start_date'),
        'end_date' => request('end_date'),
    ];
}

```

```

};

if ($searchParams['email']) {
    $query->where('email', 'LIKE', '%' . substr($searchParams['email'], 1, -1) . '%');
}

if ($searchParams['order_status']) {
    $query->where('order_status', $searchParams['order_status']);
}

if ($searchParams['payment_status']) {
    $query->where('payment_status', $searchParams['payment_status']);
}

if ($searchParams['editor']) {
    $editor = Editor::where('name', 'LIKE', '%' . $searchParams['editor'] . '%')->first();
    if ($editor) {
        ↓
        $query->where('editor', $editor->id);
    } else {
        // If no editor is found, set the query to return no results
        $query->whereNull('editor');
    }
}

if ($searchParams['start_date']) {
    $end_date = $searchParams['end_date'] ?: $searchParams['start_date'];
    $query->whereBetween('created_at', [$searchParams['start_date'], $end_date]);
}

$paginate = request('paginate', 10);
$orders = $query->paginate($paginate);

$orders->getCollection()->transform(function ($order) {
    $order->category = Category::withTrashed()->Find($order->category_id);
    $order->editor = Editor::withTrashed()->find($order->editors_id);
    $order->styles = Style::withTrashed()->whereIn('id', json_decode($order->styles_ar));
    return $order;
});

return response()->json([
    'data' => $orders
]);
}

```

[Get citation](#)

This updated function now searches for the `editors_name` and maps it to the `editor_id` before applying the filter to the `Order` query.

[Get citation](#)

Please update this code snippet here, this should also find it from with trashed values,

```

if ($searchParams['editor']) {
    $editor = Editor::where('name', 'LIKE', '%' . $searchParams['editor'] . '%')->first();
    if ($editor) {
        ↓
        $query->where('editor', $editor->id);
    } else {
        // If no editor is found, set the query to return no results
        $query->whereNull('editor');
    }
}

```

Just update this snippet and show me

Here is the updated snippet to include `withTrashed()` for the `Editor` model:

```

php
if ($searchParams['editor']) {
    $editor = Editor::withTrashed()->where('name', 'LIKE', '%' . $searchParams['editor'] . '%')->first();
    if ($editor) {
        ↓
        $query->where('editor', $editor->id);
    } else {
        // If no editor is found, set the query to return no results
        $query->whereNull('editor');
    }
}

```

[Get citation](#)

Please fix the code, It will be output in the "desc" order

Here's the updated code snippet with the search query results ordered in descending order:

```

php
public function search()
{
    $query = Order::query();

    $searchParams = [
        'email' => request('email'),
        'order_status' => request('order_status'),
        'payment_status' => request('payment_status'),
        'editor' => request('editor'),
        'start_date' => request('start_date'),
        'end_date' => request('end_date'),
    ];
}

```

```

if ($searchParams['email']) {
    $query->where('email', 'LIKE', '%' . substr($searchParams['email'], 1, -1) . '%');
}

if ($searchParams['order_status']) {
    $query->where('order_status', $searchParams['order_status']);
}

if ($searchParams['payment_status']) {
    $query->where('payment_status', $searchParams['payment_status']);
}

if ($searchParams['editor']) {
    $editor = Editor::withTrashed()->where('name', 'LIKE', '%' . $searchParams['editor']);
    if ($editor) {
        $query->where('editor', $editor->id);
    } else {
        // If no editor is found, set the query to return no results
        $query->whereNull('editor');
    }
}

if ($searchParams['start_date']) {
    $end_date = $searchParams['end_date'] ?: $searchParams['start_date'];
    $query->whereBetween('created_at', [$searchParams['start_date'], $end_date]);
}

$paginate = request('paginate', 10);
$orders = $query->orderBy('created_at', 'desc')->paginate($paginate);

$orders->getCollection()->transform(function ($order) {
    $order->category = Category::withTrashed()->find($order->category_id);
    $order->editor = Editor::withTrashed()->find($order->editors_id);
    $order->styles = Style::withTrashed()->whereIn('id', json_decode($order->styles_ar));
    return $order;
});

return response()->json([
    'data' => $orders
]);
}

```

 Get citation

Based on the searches and outputs, I also want to see the queries based on search

Here is a code sample,

```

$total_orders_count = Order::count();
$completed_orders_count =
Order::where('order_status','completed')->count();
$pending_orders_count = Order::where('order_status','pending')-
>count();
$cancelled_orders_count =
Order::where('order_status','cancelled')->count();
$preview_orders_count = Order::where('order_status','preview')-
>count();

'total_orders_count' => $total_orders_count,
'pending_orders_count' => $pending_orders_count,
'cancelled_orders_count' => $cancelled_orders_count,
'completed_orders_count' => $completed_orders_count,
'preview_orders_count' => $preview_orders_count,

```

Now from query I want to see the counts,

Now modify the code

 Get citation

 Here's the updated code to include the additional counts based on the search results:

 Get citation

```

php
public function search()
{
    $query = Order::query();

    $searchParams = [
        'email' => request('email'),
        'order_status' => request('order_status'),
        'payment_status' => request('payment_status'),
        'editor' => request('editor'),
        'start_date' => request('start_date'),
        'end_date' => request('end_date'),
    ];

    if ($searchParams['email']) {
        $query->where('email', 'LIKE', '%' . substr($searchParams['email'], 1, -1) . '%');
    }

    if ($searchParams['order_status']) {
        $query->where('order_status', $searchParams['order_status']);
    }

    if ($searchParams['payment_status']) {
        $query->where('payment_status', $searchParams['payment_status']);
    }

    if ($searchParams['editor']) {
        $editor = Editor::withTrashed()->where('name', 'LIKE', '%' . $searchParams['editor']);
        if ($editor) {
            $query->where('editor', $editor->id);
        } else {
            // If no editor is found, set the query to return no results
            $query->whereNull('editor');
        }
    }

    $total_orders_count = Order::count();
    $completed_orders_count =
    Order::where('order_status','completed')->count();
    $pending_orders_count = Order::where('order_status','pending')-
    >count();
    $cancelled_orders_count =
    Order::where('order_status','cancelled')->count();
    $preview_orders_count = Order::where('order_status','preview')-
    >count();

    'total_orders_count' => $total_orders_count,
    'pending_orders_count' => $pending_orders_count,
    'cancelled_orders_count' => $cancelled_orders_count,
    'completed_orders_count' => $completed_orders_count,
    'preview_orders_count' => $preview_orders_count,
}

```

 Get citation

```

        $query->whereNull('editor');
    }
}

if ($searchParams['start_date']) {
    $end_date = $searchParams['end_date'] ?: $searchParams['start_date'];
    $query->whereBetween('created_at', [$searchParams['start_date'], $end_date]);
}

$paginate = request('paginate', 10);
$orders = $query->orderBy('created_at', 'desc')->paginate($paginate);

// Get counts based on the search query
$total_orders_count = $query->count();
$completed_orders_count = (clone $query)->where('order_status', 'completed')->count();
$pending_orders_count = (clone $query)->where('order_status', 'pending')->count();
$cancelled_orders_count = (clone $query)->where('order_status', 'cancelled')->count();
$preview_orders_count = (clone $query)->where('order_status', 'preview')->count();

$orders->getCollection()->transform(function ($order) {
    $order->category = Category::withTrashed()->find($order->category_id);
    $order->editor = Editor::withTrashed()->find($order->editors_id);
    $order->styles = Style::withTrashed()->whereIn('id', json_decode($order->styles_ar));
    return $order;
});

return response()->json([
    'data' => $orders,
    'total_orders_count' => $total_orders_count,
    'pending_orders_count' => $pending_orders_count,
    'cancelled_orders_count' => $cancelled_orders_count,
    'completed_orders_count' => $completed_orders_count,
    'preview_orders_count' => $preview_orders_count,
]);
}

```

Explanation:

1. Counts Based on Search Query:

- The original `\$query` object is used to get the total count.
- Clone the `\$query` object to count the orders with specific `order_status` to avoid modifying the original query.
- The cloned queries are used to get the count for each order status ('completed', 'pending', 'cancelled', 'preview').

2. Returning the Counts:

- The counts are included in the JSON response along with the paginated orders.

Q D C Q ↴ ↵

G Get citation

Message ChatGPT

ChatGPT can make mistakes. Check important info.

Upgrade plan
Get GPT-4, DALL-E, and more