Change Email Subject Logic

```
@extends(BaseHelper::getAdminMasterLayoutTemplate(
))

@section('content')
    <x-core::form
        :url="$updateUrl"
        method="put"
    >
        <input
            type="hidden"
            name="module"
            value="{{ $pluginData['name'] }}"
        >
        <input
            type="hidden"
            name="template_file"
            value="{{ $pluginData['template_file'] }}"
        >

        <x-core-setting::section
            :title="trans('core/setting::setting.email.title')"

:description="trans('core/setting::setting.email.descriptio
n')"
        >
            @if ($emailSubject)
                <input
                    type="hidden"
                    name="email_subject_key"
                    value="{{
get_setting_email_subject_key($pluginData['type'],
$pluginData['name'], $pluginData['template_file']) }}"
                >

                <x-core::form.text-input
                    name="email_subject"

:label="trans('core/setting::setting.email.subject')"
                    :value="$emailSubject"
```

```
                data-counter="300"
            />
        @endif

        <x-core::form-group>
            <x-core::form.label for="mail-template-editor">
                {{ trans('core/setting::setting.email.content')
}}
            </x-core::form.label>

            <x-core::twig-editor

:variables="EmailHandler::getVariables($pluginData['typ
e'], $pluginData['name'], $pluginData['template_file'])"
                :functions="EmailHandler::getFunctions()"
                :value="$emailContent"
                name="email_content"
                mode="html"
            >
            </x-core::twig-editor>
        </x-core::form-group>

        @if (
            $metabox = apply_filters(
                'setting_email_template_meta_boxes',
                null,
                request()->route()->parameters()))
            <x-slot:footer>
                <div class="mt-3">
                    {!! $metabox !!}
                </div>
            </x-slot:footer>
        @endif
    </x-core-setting::section>

    <x-core-setting::section.action>
        <div class="btn-list">
            <x-core::button
                type="submit"
                color="primary"
                icon="ti ti-device-floppy"
            >
```

```
                {{ trans('core/setting::setting.save_settings') }}
            </x-core::button>
            <x-core::button
                tag="a"
                href="{{ $restoreUrl .
BaseHelper::stringify(request()->input('ref_lang')) }}"
                icon="ti ti-arrow-back-up"
                data-bb-toggle="reset-default"
            >
                {{
trans('core/setting::setting.email.reset_to_default') }}
            </x-core::button>
            <x-core::button
                tag="a"
                href="{{
route('settings.email.template.preview', ['type' =>
$pluginData['type'], 'module' => $pluginData['name'],
'template' => $pluginData['template_file'], 'ref_lang' =>
request()->input('ref_lang')]) }}"
                target="_blank"
                icon="ti ti-eye"
            >
                {{ trans('core/setting::setting.preview') }}
            </x-core::button>
        </div>
    </x-core-setting::section.action>
</x-core::form>

<x-core-setting::section class="mt-6">
    <h4>{{
trans('core/base::base.email_template.icon_variables') }}
</h4>

    @if (! empty($iconVariables =
EmailHandler::getIconVariables()))
        <x-core::table>
            <x-core::table.header>
                <x-core::table.header.cell width="50">
                    {{
trans('core/base::base.email_template.preview')  }}
                </x-core::table.header.cell>
```

```
                <x-core::table.header.cell>
                    {{
trans('core/base::base.email_template.variable')  }}
                </x-core::table.header.cell>
            </x-core::table.header>
            <x-core::table.body>
                @foreach (EmailHandler::getIconVariables()
as $key => $value)
                    <x-core::table.body.row>
                        <x-core::table.body.cell>
                            <div style="background-color:
#206bc4; border-radius: 50%; width: 50px; height: 50px;
display: flex; align-items: center; justify-content: center">
                                <img
                                    src="{{ $value }}"
                                    width="32"
                                    height="32"
                                    alt="{{ $key }}"
                                />
                            </div>
                        </x-core::table.body.cell>

                        <x-core::table.body.cell>
                            <span class="text-
danger">&#123;&#123; '{{ $key }}' | icon_url
&#125;&#125;</span>
                            <a
                                href="javascript:void(0);"
                                data-bb-toggle="clipboard"
                                data-clipboard-action="copy"
                                data-clipboard-text="&#123;&#123;
'{{ $key }}' | icon_url &#125;&#125;"
                                data-clipboard-message="{{
trans('core/table::table.copied') }}"
                                data-bs-toggle="tooltip"
                                class="text-muted text-center text-
decoration-none ms-1"
                            >
                                <span class="sr-only">{{
trans('core/table::table.copy') }}</span>
                                <x-core::icon name="ti ti-clipboard"
/>
```

```
                        </a>

                    </x-core::table.body.cell>
                </x-core::table.body.row>
            @endforeach
        </x-core::table.body>
    </x-core::table>
@else
    <x-core::alert type="warning" class="mt-4"
:title="trans('core/base::base.email_template.missing_icons')">
        <p class="mt-2">
            {!!
BaseHelper::clean(trans('core/base::base.email_template.missing_icons_description', [
                'from' => Html::tag('code',
'platform/core/base/public/images/email-icons'),
                'to' => Html::tag('code',
'public/vendor/core/core/base/images/email-icons'),
            ])) !!}
        </p>
    </x-core::alert>
@endif

<x-core::alert class="mt-4"
:title="trans('core/base::base.email_template.usage')">
    <ul class="mt-2">
        <li>
            <p>
                {!!
BaseHelper::clean(trans('core/base::base.email_template.icon_variable_usage_description', [
                    'variable' => Html::tag('code',
'&#123;&#123 \'example\' | icon_url &#125;&#125;'),
                ])) !!}
            </p>
            <pre><code>&lt;img src="&#123;&#123
'example' | icon_url &#125;&#125;" class="bb-va-middle"
width="40" height="40" alt="Icon"&gt;</code></pre>
        </li>
        <li>
            {!! BaseHelper::clean(trans(
```

```
'core/base::base.email_template.add_more_icon_description',
                    ['path' => Html::tag('code', 'public/vendor/core/core/base/images/email-icons')]
                ) !!}
            </li>
        </ul>
    </x-core::alert>
</x-core-setting::section>

<x-core::modal.action
    id="reset-template-to-default-modal"
    type="warning"

:title="trans('core/setting::setting.email.confirm_reset')"

:description="trans('core/setting::setting.email.confirm_message')"
    :submit-button-attrs="['id' => 'reset-template-to-default-button']"
    :submit-button-label="trans('core/setting::setting.email.continue')"
/>
@stop
```

jer e i wanted to chnage emailSubject how should i chnage this i need to add condition based on tiotlle and chnage email sobject accoding to
that how should i chnage this $emailSubject variable update ot

To update the `$emailSubject` variable based on a condition, you can add a conditional check to dynamically set the value of `$emailSubject`. For example, if you want to change the subject based on the title, you could do something like this:

1. Check the title or any other condition (you can use an `if` statement or `switch` based on your requirement).
2. Set `$emailSubject` accordingly.

Here's an example of how you might modify it:

Step-by-step Implementation:

1. Modify the Controller or Data Binding Logic (If necessary)

First, ensure that you are passing the right title or condition for setting the `$emailSubject`. This could be based on the `title` or any variable you'd like to use.

For example, in your controller or logic before the form is passed, you might want to set the `$emailSubject` conditionally:

```php
$emailSubject = ''; // Initialize the email subject variable

// Assuming you want to base it on a title, such as $pluginData['name'] or any custom condition
if ($pluginData['name'] === 'some_condition') {
    $emailSubject = 'Your Custom Subject for This Condition';
} elseif ($pluginData['name'] === 'another_condition') {
    $emailSubject = 'Another Custom Subject';
} else {
    $emailSubject = 'Default Email Subject'; // Fallback if no condition matches
}
```

2. Update the Blade View

Now, within the Blade view, you can add the dynamic `$emailSubject` to your form input field:

```php
@if ($emailSubject)
    <input
        type="hidden"
        name="email_subject_key"
        value="{{ get_setting_email_subject_key($pluginData['type'], $pluginData['name'], $pluginData['template_file']) }}"
    >

    <x-core::form.text-input
        name="email_subject"
        :label="trans('core/setting::setting.email.subject')"
        :value="$emailSubject"
        data-counter="300"
```

```
    />
@endif
```

With this setup, `$emailSubject` will change dynamically based on the condition you define before the form is rendered.

Explanation:

- **Conditional logic**: The `$emailSubject` variable is conditionally assigned before rendering the form. You can define any conditions (e.g., based on the `title`, `pluginData`, etc.).
- **Blade Template**: The `x-core::form.text-input` will then show the correct subject based on the `$emailSubject` value.
- **Default Fallback**: You can also define a fallback subject in case no conditions are met.

Example with a `switch` for multiple conditions:

php

```php
switch ($pluginData['name']) {
    case 'newsletter':
        $emailSubject = 'Your Newsletter Updates';
        break;
    case 'order_confirmation':
        $emailSubject = 'Order Confirmation - Thank You!';
        break;
    default:
        $emailSubject = 'Default Email Subject';
        break;
}
```

Let me know if you need further details on this!

@extends(BaseHelper::getAdminMasterLayoutTemplate(
))

@section('content')
  <x-core::form
    :url="$updateUrl"
    method="put"

```
    >
        <input
            type="hidden"
            name="module"
            value="{{ $pluginData['name'] }}"
        >
        <input
            type="hidden"
            name="template_file"
            value="{{ $pluginData['template_file'] }}"
        >

        <x-core-setting::section
            :title="trans('core/setting::setting.email.title')"

:description="trans('core/setting::setting.email.descriptio
n')"
        >
            @if ($emailSubject)
                <input
                    type="hidden"
                    name="email_subject_key"
                    value="{{
get_setting_email_subject_key($pluginData['type'],
$pluginData['name'], $pluginData['template_file']) }}"
                >

                <x-core::form.text-input
                    name="email_subject"

:label="trans('core/setting::setting.email.subject')"
                    :value="$emailSubject"
                    data-counter="300"
                />
            @endif

            <x-core::form-group>
                <x-core::form.label for="mail-template-editor">
                    {{ trans('core/setting::setting.email.content')
}}
                </x-core::form.label>
```

```
                <x-core::twig-editor

:variables="EmailHandler::getVariables($pluginData['type'], $pluginData['name'], $pluginData['template_file'])"
                    :functions="EmailHandler::getFunctions()"
                    :value="$emailContent"
                    name="email_content"
                    mode="html"
                >
                </x-core::twig-editor>
            </x-core::form-group>

            @if (
                $metabox = apply_filters(
                    'setting_email_template_meta_boxes',
                    null,
                    request()->route()->parameters()))
                <x-slot:footer>
                    <div class="mt-3">
                        {!! $metabox !!}
                    </div>
                </x-slot:footer>
            @endif
        </x-core-setting::section>

        <x-core-setting::section.action>
            <div class="btn-list">
                <x-core::button
                    type="submit"
                    color="primary"
                    icon="ti ti-device-floppy"
                >
                    {{ trans('core/setting::setting.save_settings') }}
                </x-core::button>
                <x-core::button
                    tag="a"
                    href="{{ $restoreUrl . BaseHelper::stringify(request()->input('ref_lang')) }}"
                    icon="ti ti-arrow-back-up"
                    data-bb-toggle="reset-default"
                >
                    {{
```

```
trans('core/setting::setting.email.reset_to_default') }}
            </x-core::button>
            <x-core::button
                tag="a"
                href="{{
route('settings.email.template.preview', ['type' =>
$pluginData['type'], 'module' => $pluginData['name'],
'template' => $pluginData['template_file'], 'ref_lang' =>
request()->input('ref_lang')]) }}"
                target="_blank"
                icon="ti ti-eye"
            >
                {{ trans('core/setting::setting.preview') }}
            </x-core::button>
        </div>
    </x-core-setting::section.action>
</x-core::form>

<x-core-setting::section class="mt-6">
    <h4>{{
trans('core/base::base.email_template.icon_variables') }}
</h4>

    @if (! empty($iconVariables =
EmailHandler::getIconVariables()))
        <x-core::table>
            <x-core::table.header>
                <x-core::table.header.cell width="50">
                    {{
trans('core/base::base.email_template.preview')  }}
                </x-core::table.header.cell>

                <x-core::table.header.cell>
                    {{
trans('core/base::base.email_template.variable')  }}
                </x-core::table.header.cell>
            </x-core::table.header>
            <x-core::table.body>
                @foreach (EmailHandler::getIconVariables()
as $key => $value)
                    <x-core::table.body.row>
                        <x-core::table.body.cell>
```

```
                        <div style="background-color:
#206bc4; border-radius: 50%; width: 50px; height: 50px;
display: flex; align-items: center; justify-content: center">
                            <img
                                src="{{ $value }}"
                                width="32"
                                height="32"
                                alt="{{ $key }}"
                            />
                        </div>
                    </x-core::table.body.cell>

                    <x-core::table.body.cell>
                        <span class="text-
danger">&amp;#123;&amp;#123; '{{ $key }}' | icon_url
&amp;#125;&amp;#125;</span>
                        <a
                            href="javascript:void(0);"
                            data-bb-toggle="clipboard"
                            data-clipboard-action="copy"
                            data-clipboard-text="&amp;#123;&amp;#123;
'{{ $key }}' | icon_url &amp;#125;&amp;#125;"
                            data-clipboard-message="{{
trans('core/table::table.copied') }}"
                            data-bs-toggle="tooltip"
                            class="text-muted text-center text-
decoration-none ms-1"
                        >
                            <span class="sr-only">{{
trans('core/table::table.copy') }}</span>
                            <x-core::icon name="ti ti-clipboard"
/>
                        </a>

                    </x-core::table.body.cell>
                </x-core::table.body.row>
            @endforeach
        </x-core::table.body>
    </x-core::table>
@else
    <x-core::alert type="warning" class="mt-4"
:title="trans('core/base::base.email_template.missing_ico
```

```
ns')">
            <p class="mt-2">
                {!!
BaseHelper::clean(trans('core/base::base.email_template
.missing_icons_description', [
                    'from' => Html::tag('code',
'platform/core/base/public/images/email-icons'),
                    'to' => Html::tag('code',
'public/vendor/core/core/base/images/email-icons'),
                ])) !!}
            </p>
        </x-core::alert>
    @endif

    <x-core::alert class="mt-4"
:title="trans('core/base::base.email_template.usage')">
        <ul class="mt-2">
            <li>
                <p>
                    {!!
BaseHelper::clean(trans('core/base::base.email_template
.icon_variable_usage_description', [
                        'variable' => Html::tag('code',
'&#123;&#123 \'example\' | icon_url &#125;&#125;'),
                    ])) !!}
                </p>
                <pre><code>&lt;img src="&#123;&#123
'example' | icon_url &#125;&#125;" class="bb-va-middle"
width="40" height="40" alt="Icon"&gt;</code></pre>
            </li>
            <li>
                {!! BaseHelper::clean(trans(

'core/base::base.email_template.add_more_icon_descrip
tion',
                    ['path' => Html::tag('code',
'public/vendor/core/core/base/images/email-icons')]
                )) !!}
            </li>
        </ul>
    </x-core::alert>
</x-core-setting::section>
```

```
    <x-core::modal.action
        id="reset-template-to-default-modal"
        type="warning"

:title="trans('core/setting::setting.email.confirm_reset')"

:description="trans('core/setting::setting.email.confirm_
message')"
        :submit-button-attrs="['id' => 'reset-template-to-
default-button']"
        :submit-button-
label="trans('core/setting::setting.email.continue')"
    />
@stop
```

jer e i wanted to chnage emailSubject how should i chnage this i need to add condition based on tiotlle and chnage email sobject accoding to
that how should i chnage this $emailSubject variable update add conidtiomn like that that form hidden filed that name type_of_form
should have "international-patients" then just show this text in that subject otherwise "book-an-appointment" that filed have text then
just show that modify condition accoding to that

To implement the conditional logic you're asking for, where the `$emailSubject` changes based on the value of the `type_of_form` hidden field, you can do the following:

Steps:

1. **Check for the `type_of_form` hidden field value**.
2. **Set `$emailSubject` based on that condition**.

You can modify the `$emailSubject` setting logic in your controller or data-binding logic before rendering the form. Here's how you can achieve that:

1. Add a hidden field in the form for `type_of_form`

Add the `type_of_form` hidden field to your form, which will hold the form's type (`international-patients` or `book-an-appointment`).

For example, add the following hidden field to the form:

```html
<input type="hidden" name="type_of_form" value="{{ old('type_of_form', 'international-patients') }}">
```

This field will have a default value, but it can be changed dynamically when needed.

2. Conditional Logic for `$emailSubject`

Now, you can modify the `$emailSubject` based on the `type_of_form` value. You can handle this logic in your controller, or you can even do it directly in the Blade view.

In the Controller (Recommended)

Before rendering the view, you'll check the value of `type_of_form` and update `$emailSubject` accordingly.

For example:

```php
$typeOfForm = request()->input('type_of_form', 'international-patients'); // Default to 'international-patients'

if ($typeOfForm === 'international-patients') {
    $emailSubject = 'Subject for International Patients';
} else {
    $emailSubject = 'Subject for Book an Appointment';
}

// Pass this to the view
return view('your.view', compact('emailSubject', 'typeOfForm'));
```

In the Blade View

In the Blade view, you can also implement the logic directly:

```php
@php
    $typeOfForm = request()->input('type_of_form', 'international-patients'); // Default to 'international-patients'
```

```
    if ($typeOfForm === 'international-patients') {
        $emailSubject = 'Subject for International Patients';
    } else {
        $emailSubject = 'Subject for Book an Appointment';
    }
@endphp
```

3. Rendering the Form

Now, in your form, you can use the `$emailSubject` variable that was conditionally set:

```php
@if ($emailSubject)
    <input
        type="hidden"
        name="email_subject_key"
        value="{{ get_setting_email_subject_key($pluginData['type'], $pluginData['name'], $pluginData['template_file']) }}"
    >

    <x-core::form.text-input
        name="email_subject"
        :label="trans('core/setting::setting.email.subject')"
        :value="$emailSubject"
        data-counter="300"
    />
@endif
```

Summary of Changes:

1. **Hidden Field (`type_of_form`)**: This field determines whether the subject should be for "international-patients" or "book-an-appointment".
2. **Conditional Logic**: The `$emailSubject` is set based on the value of `type_of_form`.
3. **Passing the Subject to the Form**: The `email_subject` input field is populated with the conditional `$emailSubject` value.

Example Flow:

1. **If `type_of_form` is `international-patients`**, the `$emailSubject` will be set to "Subject for International Patients".

2. **If** `type_of_form` **is** `book-an-appointment`, the `$emailSubject` will be set to "Subject for Book an Appointment".

Let me know if you need further assistance!

```
<?php

namespace Botble\Contact\Forms\Fronts;

use Botble\Base\Facades\Html;
use Botble\Base\Forms\FieldOptions\ButtonFieldOption;
use
Botble\Base\Forms\FieldOptions\CheckboxFieldOption;
use Botble\Base\Forms\FieldOptions\HtmlFieldOption;
use Botble\Base\Forms\FieldOptions\InputFieldOption;
use
Botble\Base\Forms\FieldOptions\NumberFieldOption;
use Botble\Base\Forms\FieldOptions\RadioFieldOption;
use Botble\Base\Forms\FieldOptions\SelectFieldOption;
use
Botble\Base\Forms\FieldOptions\TextareaFieldOption;
use Botble\Base\Forms\FieldOptions\TextFieldOption;
use Botble\Base\Forms\Fields\DateField;
use Botble\Base\Forms\Fields\DatetimeField;
use Botble\Base\Forms\Fields\EmailField;
use Botble\Base\Forms\Fields\HtmlField;
use Botble\Base\Forms\Fields\NumberField;
use Botble\Base\Forms\Fields\OnOffCheckboxField;
use Botble\Base\Forms\Fields\RadioField;
use Botble\Base\Forms\Fields\SelectField;
use Botble\Base\Forms\Fields\TextareaField;
use Botble\Base\Forms\Fields\TextField;
use Botble\Base\Forms\Fields\TimeField;
use Botble\Contact\Enums\CustomFieldType;
use Botble\Contact\Http\Requests\ContactRequest;
use Botble\Contact\Models\Contact;
use Botble\Contact\Models\CustomField;
use Botble\Theme\Facades\Theme;
use Botble\Theme\FormFront;
use Closure;
use Illuminate\Contracts\Support\Arrayable;
use Illuminate\Database\Eloquent\Collection;
```

```
use Illuminate\Support\Arr;
use Throwable;

class ContactForm extends FormFront
{
    protected string $errorBag = 'contact';

    protected ?string $formInputWrapperClass = 'contact-
form-group';

    protected ?string $formInputClass = 'contact-form-
input';

    public static function formTitle(): string
    {
        return
trans('plugins/contact::contact.contact_form');
    }

    public function setup(): void
    {
        $data = $this->getModel();

        try {
            $displayFields = array_filter(explode(',', (string)
Arr::get($data, 'display_fields'))) ?: ['phone', 'email',
'address', 'subject'];
        } catch (Throwable) {
            $displayFields = ['phone', 'email', 'address',
'subject'];
        }

        try {
            $mandatoryFields = array_filter(explode(',', (string)
Arr::get($data, 'mandatory_fields'))) ?: ['email'];
        } catch (Throwable) {
            $mandatoryFields = ['email'];
        }

        $this
            ->contentOnly()
            ->model(Contact::class)
```

```
            ->setUrl(route('public.send.contact'))
            ->setValidatorClass(ContactRequest::class)
            ->setFormOption('class', 'contact-form')
            ->add(
                'filters_before_form',
                HtmlField::class,
                HtmlFieldOption::make()
                    ->content(apply_filters('pre_contact_form',
null))
            )
            ->add(
                'required_fields',
                'hidden',
                TextFieldOption::make()
                    ->value(Arr::get($data, 'mandatory_fields'))
            )
            ->add(
                'display_fields',
                'hidden',
                TextFieldOption::make()
                    ->value(Arr::get($data, 'display_fields'))
            )
            ->addRowWrapper('form_wrapper', function (self
$form) use ($displayFields, $mandatoryFields): void {
                $customFields = CustomField::query()
                    ->wherePublished()
                    ->orderBy('order')
                    ->get();

                $form
                    ->addColumnWrapper('name_wrapper',
function (self $form): void {
                        $form
                            ->add(
                                'name',
                                TextField::class,
                                TextFieldOption::make()
                                    ->required()
                                    ->label(__('Name'))
                                    ->placeholder(__('Your Name'))
                                    ->wrapperAttributes(['class' => $this-
>formInputWrapperClass])
```

```
                        ->cssClass($this->formInputClass)
                        ->maxLength(-1)
                );
            })
            ->when(in_array('email', $displayFields),
function (self $form) use ($mandatoryFields): void {
                $form
                    ->addColumnWrapper('email_wrapper',
function (self $form) use ($mandatoryFields): void {
                        $form
                            ->add(
                                'email',
                                EmailField::class,
                                TextFieldOption::make()
                                    ->when(in_array('email',
$mandatoryFields), function (TextFieldOption $option):
void {
                                        $option->required();
                                    })
                                    ->label(__('Email'))
                                    ->placeholder(__('Your Email'))
                                    ->wrapperAttributes(['class' =>
$this->formInputWrapperClass])
                                    ->cssClass($this-
>formInputClass)
                                    ->maxLength(-1)
                            );
                    });
            })
            ->when(in_array('address', $displayFields),
function (self $form) use ($mandatoryFields): void {
                $form-
>addColumnWrapper('address_wrapper', function (self
$form) use ($mandatoryFields): void {
                    $form
                        ->add(
                            'address',
                            TextField::class,
                            TextFieldOption::make()
                                ->when(in_array('address',
$mandatoryFields), function (TextFieldOption $option):
void {
```

```
                                $option->required();
                            })
                            ->label(__('Address'))
                            ->placeholder(__('Your Address'))
                            ->wrapperAttributes(['class' =>
$this->formInputWrapperClass])
                            ->cssClass($this->formInputClass)
                            ->maxLength(-1)
                    );
                });
            })
            ->when(in_array('phone', $displayFields),
function (self $form) use ($mandatoryFields): void {
                $form-
>addColumnWrapper('phone_wrapper', function (self
$form) use ($mandatoryFields): void {
                    $form
                        ->add(
                            'phone',
                            TextField::class,
                            TextFieldOption::make()
                                ->when(in_array('phone',
$mandatoryFields), function (TextFieldOption $option):
void {
                                    $option->required();
                                })
                                ->label(__('Phone'))
                                ->placeholder(__('Your Phone'))
                                ->wrapperAttributes(['class' =>
$this->formInputWrapperClass])
                                ->cssClass($this->formInputClass)
                                ->maxLength(-1)
                        );
                });
            })
            ->when(in_array('subject', $displayFields),
function (self $form) use ($mandatoryFields): void {
                $form-
>addColumnWrapper('subject_wrapper', function (self
$form) use ($mandatoryFields): void {
                    $form->add(
                        'subject',
```

```
                        TextField::class,
                        TextFieldOption::make()
                            ->when(in_array('subject',
$mandatoryFields), function (TextFieldOption $option):
void {
                                $option->required();
                            })
                            ->label(__('Subject'))
                            ->placeholder(__('Subject'))
                            ->wrapperAttributes(['class' => $this-
>formInputWrapperClass])
                            ->cssClass($this->formInputClass)
                            ->maxLength(-1)
                    );
                }, 12);
            })
            ->when($customFields, function
(ContactForm $form, Collection $customFields): void {
                foreach ($customFields as $customField) {
                    $options = $customField->options()-
>select('id', 'label', 'value')->get()->mapWithKeys(function
($option) {
                        return [$option->value => $option-
>label];
                    })->all();

                    $fieldOptions = match ($customField-
>type->getValue()) {
                        CustomFieldType::NUMBER =>
NumberFieldOption::make()
                            ->when($customField->placeholder,
function (InputFieldOption $options, string
$placeholder): void {
                                $options-
>placeholder($placeholder);
                            }),
                        CustomFieldType::DROPDOWN =>
SelectFieldOption::make()
                            ->when($customField->placeholder,
function (SelectFieldOption $fieldOptions, string
$placeholder) use ($options): void {
                                $fieldOptions->choices(['' =>
```

```
$placeholder, ...$options]);
                        }, function (SelectFieldOption
$fieldOptions) use ($options): void {
                            $fieldOptions->choices($options);
                        }),
                    CustomFieldType::CHECKBOX =>
CheckboxFieldOption::make(),
                    CustomFieldType::RADIO =>
RadioFieldOption::make()->choices($options),
                    default => TextFieldOption::make()
                        ->wrapperAttributes(['class' => $this-
>formInputWrapperClass])
                        ->cssClass($this->formInputClass)
                        ->placeholder($customField-
>placeholder ?: $customField->name)
                };

                $field = match ($customField->type-
>getValue()) {
                    CustomFieldType::NUMBER =>
NumberField::class,
                    CustomFieldType::TEXTAREA =>
TextareaField::class,
                    CustomFieldType::DROPDOWN =>
SelectField::class,
                    CustomFieldType::CHECKBOX =>
OnOffCheckboxField::class,
                    CustomFieldType::RADIO =>
RadioField::class,
                    CustomFieldType::DATE =>
DateField::class,
                    CustomFieldType::DATETIME =>
DatetimeField::class,
                    CustomFieldType::TIME =>
TimeField::class,
                    default => TextField::class,
                };

                $form-
>addColumnWrapper("custom_field_{$customField-
>id}_wrapper", function (self $form) use ($customField,
$field, $fieldOptions): void {
```

```
                        $form->add(

"contact_custom_fields[$customField->id]",
                            $field,
                            $fieldOptions
                                ->label($customField->name)
                                ->required($customField-
>required)
                                ->wrapperAttributes(['class' =>
$this->formInputWrapperClass])
                            );
                        }, 12);
                    }
                });
            })
            ->addRowWrapper(
                'content',
                function (self $form): void {
                    $form->addColumnWrapper(
                        'content',
                        function (self $form): void {
                            $form->add(
                                'content',
                                TextareaField::class,
                                TextareaFieldOption::make()
                                    ->required()
                                    ->label(__('Notes'))
                                    ->placeholder(__('Write your
message here'))
                                    ->wrapperAttributes(['class' => $this-
>formInputWrapperClass])
                                    ->cssClass($this->formInputClass)
                                    ->rows(5)
                                    ->maxLength(-1)
                            );
                        },
                        12
                    );
                }
            )
            ->add(
                'filters_after_form',
```

```
            HtmlField::class,
            HtmlFieldOption::make()
                ->content(apply_filters('after_contact_form',
null))
        )
        ->add(
            'agree_terms_and_policy',
            OnOffCheckboxField::class,
            CheckboxFieldOption::make()
                ->label(
                    ($privacyPolicyUrl =
Theme::termAndPrivacyPolicyUrl())
                        ? __('I agree to the :link', ['link' =>
Html::link($privacyPolicyUrl, __('Terms and Privacy
Policy'), attributes: ['class' => 'text-decoration-underline',
'target' => '_blank'])])
                        : __('I agree to the Terms and Privacy
Policy')
                )
        )
        ->addWrappedField(
            'submit',
            'submit',
            ButtonFieldOption::make()
                ->cssClass('contact-button')
                ->label(__('Send'))
        )
        ->addWrappedField(
            'messages',
            HtmlField::class,
            HtmlFieldOption::make()
                ->content(<<<'HTML'
                    <div class="contact-message contact-
success-message" style="display: none"></div>
                    <div class="contact-message contact-error-
message" style="display: none"></div>
                HTML)
        );
    }
```

this would be or form where we want to pass hidden file name type_of_form and pass value as "International

form" while
sending emils

To pass a hidden input field named `type_of_form` with the value `International form` in the form, and then use that information to modify the email subject based on the condition, follow these steps:

1. **Add the Hidden Field to the Form:** Inside your form structure, add a hidden input field with the name `type_of_form` and a default value of `International form`.
2. **Modify the Logic for Email Subject:** You'll need to update your logic where you're setting the `$emailSubject` to check the value of the `type_of_form` field and adjust the subject based on the form type.

1. Add the Hidden Input Field

Add the hidden input field `type_of_form` in the `setup()` method in your form like so:

```php
$this->add(
    'type_of_form',
    'hidden',
    TextFieldOption::make()
        ->value('International form') // This will set the default value to "International form"
);
```

This will make sure that the form contains the hidden input with the value `"International form"`.

2. Modify the Email Subject Based on `type_of_form`

Now, where you're generating the `$emailSubject`, you can conditionally modify it based on the value of `type_of_form`.

Let's say this is where the `$emailSubject` is defined (possibly in a controller or a method that handles email sending):

```php
// Check the type_of_form value and modify the subject accordingly
$typeOfForm = request()->input('type_of_form'); // This will fetch the value of the hidden field
```

```
if ($typeOfForm === 'International form') {
    $emailSubject = "International Patient Inquiry"; // Custom subject for
international patients
} else {
    $emailSubject = "Book an Appointment"; // Default subject for other cases
}
```

Example:

If you want to implement this logic in your form's submit handler, you might end up with something like this:

php

```
// Assuming you're sending an email after the form is submitted:
$emailSubject = 'Default Subject'; // Set a default subject

// Get the form type value
$typeOfForm = request()->input('type_of_form');

// Modify the subject based on the form type
if ($typeOfForm === 'International form') {
    $emailSubject = "International Patient Inquiry";
} else {
    $emailSubject = "Book an Appointment";
}

// Continue with sending the email...
Mail::to($recipient)->send(new ContactFormMail($emailSubject, $emailContent));
// Use the modified subject
```

Complete Example in Context

Here's how it fits within your current form setup and email sending logic:

php

```
// Add the hidden field 'type_of_form' to the form
$this->add(
    'type_of_form',
    'hidden',
    TextFieldOption::make()
```

```
        ->value('International form') // You can adjust this dynamically based
on your conditions
);

// Modify email subject based on form type
$typeOfForm = request()->input('type_of_form'); // Get value of 'type_of_form'
from the form

if ($typeOfForm === 'International form') {
    $emailSubject = "International Patient Inquiry"; // For International Form
} else {
    $emailSubject = "Book an Appointment"; // Default for others
}

// Proceed with email sending logic
Mail::to($recipient)->send(new ContactFormMail($emailSubject, $emailContent));
// Pass the dynamic subject to the mail
```

Final Output:

- **Email Subject for "International form"**: The subject will be `"International Patient Inquiry"`.
- **Email Subject for other forms**: The subject will be `"Book an Appointment"`.

This solution ensures that when the user submits the form with `type_of_form = "International form"`, the email subject changes accordingly.