Báo Cáo Kỹ Thuật: DiscountForm Component

Table of Contents

1. Giới thiệu

Tổng quan về tính năng/module

Component `DiscountForm` là một thành phần quan trọng trong quy trình thanh toán của nền tảng thương mại điện tử giày dép. Component này cho phép người dùng nhập và áp dụng mã giảm giá vào đơn hàng, cũng như hiển thị thông tin về mã giảm giá đã được áp dụng và cho phép xóa mã giảm giá khi cần thiết.

Mục tiêu và phạm vi

Báo cáo này tập trung vào phân tích cấu trúc, chức năng và cách triển khai của component `DiscountForm`. Mục tiêu là làm rõ:

- Cấu trúc và các thành phần của form nhập mã giảm giá
- Cách xử lý validation dữ liệu đầu vào
- Phương thức áp dụng và xóa mã giảm giá
- Logic hiển thị UI trong các trạng thái khác nhau

2. Cấu trúc tổng quan

Danh sách thành phần chính

- **Form Input**: Phần nhập liệu cho mã giảm giá, được xây dựng bằng React Hook Form và Zod validation
- **Applied Discount Display**: Hiển thị thông tin mã giảm giá đã được áp dụng
- **Action Buttons**: Các nút để áp dụng hoặc xóa mã giảm giá

Mối quan hệ giữa các thành phần

Component `DiscountForm` sử dụng điều kiện để chuyển đổi giữa hai trạng thái hiển thị:

1. **Trạng thái nhập mã giảm giá**: Hiển thị form nhập mã và nút áp dụng khi chưa có mã giảm giá được áp dụng
2. **Trạng thái đã áp dụng mã giảm giá**: Hiển thị thông tin mã giảm giá và nút xóa khi đã có mã giảm giá được áp dụng

Component này tương tác với component cha thông qua props callback, gửi thông tin về mã giảm giá lên component cha để xử lý logic giảm giá.

3. Kiểu dữ liệu và định nghĩa

Interface và Type

```
interface DiscountFormProps {
  onApplyDiscount: (discountCode: string) => void;
  appliedDiscount: DiscountCode | null;
  isLoading?: boolean;
}
```

Giải thích:

- `onApplyDiscount`: Function callback khi người dùng áp dụng hoặc xóa mã giảm giá
- `appliedDiscount`: Đối tượng chứa thông tin về mã giảm giá đã được áp dụng (hoặc `null` nếu chưa có)
- `isLoading`: Trạng thái loading để hiển thị UI phù hợp (tham số tùy chọn, mặc định là `false`)

Props và State

Component này nhận các props từ component cha và quản lý form state thông qua React Hook Form:

```
// Định nghĩa dữ liệu form
import { discountSchema, DiscountFormValues } from '@/lib/form-validation';

// Kiểu dữ liệu của mã giảm giá
import { DiscountCode } from '@/types/checkout';
```

Schema và quy tắc xác thực

Component sử dụng Zod để xác thực dữ liệu đầu vào thông qua `discountSchema`, được import từ module `@/lib/form-validation`. Schema này định nghĩa các quy tắc xác thực cho trường `discountCode`.

4. Chi tiết về từng thành phần

Imports

```
import React from 'react';
import { useForm } from 'react-hook-form';
import { zodResolver } from '@hookform/resolvers/zod';
import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { DiscountCode } from '@/types/checkout';
import { discountSchema, DiscountFormValues } from '@/lib/form-validation';
import {
  Form,
  FormControl,
```

```
  FormField,
  FormItem,
  FormMessage,
} from '@/components/ui/form';
```

Giải thích:

- `react-hook-form`: Thư viện quản lý form trong React
- `@hookform/resolvers/zod`: Tích hợp Zod validation với React Hook Form
- Các components UI từ thư viện UI nội bộ (button, input, form elements)
- Types và schema validation (`DiscountCode`, `discountSchema`, `DiscountFormValues`)

Tham số đầu vào và đầu ra

Đầu vào (Input)

Component `DiscountForm` nhận các tham số đầu vào sau:

Tham số	Kiểu dữ liệu	Mô tả
`onApplyDiscount`	`(discountCode: string) ⇒ void`	Hàm callback được gọi khi người dùng áp dụng hoặc xóa mã giảm giá
`appliedDiscount`	`DiscountCode | null`	Thông tin về mã giảm giá đã được áp dụng (nếu có)
`isLoading`	`boolean (optional)`	Trạng thái loading, mặc định là `false`

Chi tiết về `DiscountCode`:

```
interface DiscountCode {
  code: string;
  discountType: 'percentage' | 'fixed';
  discountAmount: number;
  isValid: boolean;
}
```

Đầu ra (Output)

Component `DiscountForm` không trả về dữ liệu trực tiếp mà gọi hàm callback `onApplyDiscount` với:

Tham số	Kiểu dữ liệu	Mô tả
`discountCode`	`string`	Mã giảm giá người dùng nhập (hoặc chuỗi rỗng nếu xóa mã)

Kết quả tương tác:

- Khi người dùng áp dụng mã giảm giá: Gọi `onApplyDiscount(code)` với mã được nhập
- Khi người dùng xóa mã giảm giá: Gọi `onApplyDiscount('')` với chuỗi rỗng
- Component cha xử lý việc kiểm tra tính hợp lệ của mã giảm giá và cập nhật lại `appliedDiscount`

Khởi tạo Form

```
const form = useForm<DiscountFormValues>({
  resolver: zodResolver(discountSchema),
  defaultValues: {
    discountCode: ''
  }
});
```

Giải thích:

- Sử dụng hook `useForm` để khởi tạo và quản lý form
- Tích hợp `zodResolver` để xác thực dữ liệu theo schema đã định nghĩa
- Thiết lập giá trị mặc định cho trường `discountCode` là chuỗi rỗng

Xử lý Form Submission

```
const onSubmit = (data: DiscountFormValues) => {
  onApplyDiscount(data.discountCode);
};
```

Giải thích:

- Hàm `onSubmit` được gọi khi form được submit
- Gọi callback function `onApplyDiscount` và truyền mã giảm giá được nhập vào
- Component cha sẽ xử lý logic kiểm tra và áp dụng mã giảm giá

Xử lý Xóa Mã Giảm Giá

```
const handleRemoveDiscount = () => {
  onApplyDiscount('');
  form.reset();
};
```

Giải thích:

- Hàm `handleRemoveDiscount` được gọi khi người dùng nhấn nút Xóa
- Gọi callback function `onApplyDiscount` với chuỗi rỗng để báo hiệu xóa mã giảm giá

- Reset form về trạng thái ban đầu === Render UI

Cấu trúc tổng quát

```
return (
  <div className="border rounded-md p-4 mb-4">
    <h3 className="text-sm font-medium mb-3">Mã gi□m giá</h3>

    {appliedDiscount && appliedDiscount.isValid ? (
      // UI hi□n th□ khi đã áp d□ng mã gi□m giá
    ) : (
      // UI form nh□p mã gi□m giá
    )}
  </div>
);
```

Giải thích:

- Container chung với viền bo tròn và padding
- Tiêu đề "Mã giảm giá"
- Sử dụng biểu thức điều kiện để hiển thị UI phù hợp dựa trên trạng thái của `appliedDiscount`

UI hiển thị khi đã áp dụng mã giảm giá

```
<div>
  <div className="flex items-center justify-between bg-green-50 border border-green-
200 rounded p-2 mb-3">
    <div>
      <p className="text-sm font-medium">{appliedDiscount.code}</p>
      <p className="text-xs text-green-600">
        {appliedDiscount.discountType === 'percentage'
          ? `Gi□m ${appliedDiscount.discountAmount}%`
          : `Gi□m ${appliedDiscount.discountAmount.toLocaleString('vi-VN')}đ`}
      </p>
    </div>
    <Button
      variant="ghost"
      size="sm"
      onClick={handleRemoveDiscount}
      disabled={isLoading}
      className="h-8 text-sm"
    >
      Xóa
    </Button>
  </div>
</div>
```

Giải thích:

- Hiển thị thông tin mã giảm giá trong một box với màu nền xanh nhạt
- Hiển thị mã giảm giá (`appliedDiscount.code`)
- Hiển thị thông tin giảm giá dựa trên loại giảm giá (`percentage` hoặc giá trị cố định)
- Nút "Xóa" để xóa mã giảm giá đã áp dụng
- Nút bị vô hiệu hóa khi `isLoading` là `true`

UI form nhập mã giảm giá

```
<Form {...form}>
  <form onSubmit={form.handleSubmit(onSubmit)} className="flex gap-2">
    <div className="flex-1">
      <FormField
        control={form.control}
        name="discountCode"
        render={({ field }) => (
          <FormItem>
            <FormControl>
              <Input
                placeholder="Nh□p mã gi□m giá"
                disabled={isLoading}
                className="h-9"
                {...field}
              />
            </FormControl>
            <FormMessage className="text-xs mt-1" />
          </FormItem>
        )}
      />
    </div>
    <Button
      type="submit"
      disabled={isLoading}
      className="whitespace-nowrap h-9"
    >
      {isLoading ? 'Đang x□ lý...' : 'Áp d□ng'}
    </Button>
  </form>
</Form>
```

Giải thích:

- Sử dụng `Form` component từ thư viện UI để tích hợp với React Hook Form
- Layout flex để hiển thị input và button trên cùng một hàng
- FormField cho phép điều khiển và xác thực trường input

- Hiển thị thông báo lỗi xác thực (nếu có) thông qua `FormMessage`
- Button submit với text thay đổi dựa trên trạng thái loading
- Cả input và button đều bị vô hiệu hóa khi `isLoading` là `true`

5. Cải tiến và thay đổi đã thực hiện

Thiết kế UI thân thiện với người dùng

- **Trạng thái loading rõ ràng**: Hiển thị "Đang xử lý..." và vô hiệu hóa các controls khi đang xử lý.
- **Phản hồi trực quan**: Sử dụng màu nền xanh nhạt khi mã giảm giá được áp dụng thành công.
- **Validation form**: Tích hợp Zod để xác thực đầu vào, đảm bảo dữ liệu hợp lệ trước khi gửi.

Cấu trúc mã nguồn rõ ràng

- **Tách biệt logic và UI**: Logic xử lý form được tách biệt với phần render UI.
- **Sử dụng React Hook Form**: Tận dụng thư viện quản lý form mạnh mẽ để giảm code boilerplate.
- **Conditional rendering**: Sử dụng điều kiện để hiển thị UI phù hợp dựa trên trạng thái hiện tại.

Xử lý đa dạng loại giảm giá

Component có khả năng hiển thị cả hai loại giảm giá:

- Giảm giá theo phần trăm (%)
- Giảm giá theo giá trị cố định (VNĐ)

6. Kết luận

Tổng kết tính năng

Component `DiscountForm` cung cấp giao diện người dùng đơn giản nhưng hiệu quả để áp dụng và quản lý mã giảm giá trong quá trình thanh toán. Component được thiết kế với tính linh hoạt cao, có thể xử lý nhiều loại giảm giá khác nhau và hiển thị UI phù hợp dựa trên trạng thái hiện tại.

Đánh giá và đề xuất

Đánh giá:

- Component có cấu trúc rõ ràng, dễ bảo trì và mở rộng
- Sử dụng các thư viện hiện đại (React Hook Form, Zod) để quản lý form và validation
- UI thân thiện với người dùng với phản hồi trực quan

Đề xuất cải tiến:

- **Thêm animation**: Có thể thêm animation khi chuyển đổi giữa các trạng thái UI để tăng trải nghiệm người dùng
- **Thêm thông báo hết hạn**: Hiển thị thời gian hết hạn của mã giảm giá (nếu có)
- **Tối ưu hóa cho thiết bị di động**: Đảm bảo trải nghiệm tốt trên các thiết bị màn hình nhỏ
- **Thêm tính năng gợi ý mã giảm giá**: Có thể thêm gợi ý các mã giảm giá có sẵn hoặc phổ biến