```typescript
import React, {
  ChangeEvent,
  Dispatch,
  FormEvent,
  SetStateAction,
  useEffect,
  useState,
} from 'react';
import { Textarea } from '@/shared/components/ui/textarea';

import { Dialog, DialogContent } from '@/shared/components/ui/dialog';
import { Button } from '@/shared/components/ui/Button';
import { useToast } from '@/shared/components/ui/Toasts/use-toast';
import { useAuthContext } from '@/shared/components/ui/auth.provider';
import LoadingSpinner from '@/shared/components/loading-spinner';
import { cn } from '@/shared/utils/cn';
import { trpc } from '@/shared/utils/trpc/trpc';

interface EditReplyFormProps {
  replyId: string;
  open: boolean;
  editableContent: string;
  refetchReplies: () => void;
  setOpen: Dispatch<SetStateAction<boolean>>;
}
```

```tsx
export default function EditReplyForm({
  replyId,
  open,
  editableContent,
  setOpen,
  refetchReplies,
}: EditReplyFormProps) {
  const { user } = useAuthContext();
  const toast = useToast();
  const [content, setContent] = useState('');
  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState('');


  function handleChange(e: ChangeEvent<HTMLTextAreaElement>) {
    setContent(e.target.value);
    setError('');
  }


  const editReplyMutation = trpc.explorerOptions.editReply.useMutation();


  const handleSubmit = async (e: FormEvent) => {
    e.preventDefault();


    if (!user) {
      toast.toast({
        description: 'Log in to perform this action',
```

```
      style: { color: 'red' },
    });
    return;
  }


  if (!content) {
    toast.toast({
      description: 'Reply cannot be empty',
      style: { color: 'red' },
    });
    setError('Comment cannot be empty');
    return;
  }


  setIsLoading(true);


  editReplyMutation
    .mutateAsync({ content, replyId })
    .then(() => {
      toast.toast({
        description: 'Reply edited successfully',
        style: { color: 'green' },
      });
      setContent('');
      refetchReplies();
      setOpen(false);
```

```
      })
      .catch((err) => {
        console.error(err);
        toast.toast({
          description:
            err?.data?.message || err?.message || 'Error editing reply',
          variant: 'destructive',
        });
      })
      .finally(() => setIsLoading(false));
  };


  useEffect(() => {
    setContent(editableContent);
  }, [editableContent]);


  return (
    <Dialog open={open} onOpenChange={setOpen}>
      <DialogContent className="max-w-md flex flex-col gap-2">
        <h2 className="font-medium text-xl">Edit this reply</h2>

        <form onSubmit={handleSubmit} className="mt-5">
          <Textarea
            value={content}
            onChange={handleChange}
            cols={2}
```

```jsx
          className="border-slate-800"
        />
        <small className={cn('invisible mt-1', error ? 'visible' : '')}>
          {error}
        </small>
        <Button type="submit" variant="destructive" className="mt-8">
          <span className="mr-2">Edit Reply</span>{' '}
          {isLoading && <LoadingSpinner size={18} />}
        </Button>
      </form>
    </DialogContent>
  </Dialog>
);
}
```