```tsx
import React, { ChangeEvent, FormEvent, RefObject, useState } from 'react';
import { Textarea } from '@/shared/components/ui/textarea';
import { Button } from '@/shared/components/ui/Button';
import { trpc } from '@/shared/utils/trpc/trpc';
import { useAuthContext } from '@/shared/components/ui/auth.provider';
import { useToast } from '@/shared/components/ui/Toasts/use-toast';
import LoadingSpinner from '@/shared/components/loading-spinner';
import { cn } from '@/shared/utils/cn';

interface CommentFormProps {
  modelId: string;
  title: string;
  commentsEndRef: RefObject<HTMLDivElement>;
  refetchComments: () => void;
}

export default function CommentForm({
  modelId,
  title,
  commentsEndRef,
  refetchComments,
}: CommentFormProps) {
  const { user } = useAuthContext();

  const addComment = trpc.explorerOptions.addComment.useMutation();
```

```
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('');
}

function handleSubmit(e: FormEvent) {
  e.preventDefault();

  if (!user) {
    toast.toast({
      description: 'Log in to perform this action',
      style: { color: 'red' },
    });
    return;
  }

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

```
  setError('Comment cannot be empty');
  return;
}

setIsLoading(true);

addComment
  .mutateAsync({
    content,
    modelId,
    modelType: title,
  })
  .then(() => {
    toast.toast({
      description: 'Comment added successfully',
      style: { color: 'green' },
    });
    setContent('');
    refetchComments();

    setTimeout(() => {
      commentsEndRef.current?.scrollIntoView({ behavior: 'smooth' });
    }, 1000);
  })
  .catch((err) => {
    console.error(err);
```

```
      toast.toast({
        description:
          err?.data?.message || err?.message || 'Error adding comment',
        variant: 'destructive',
      });
    })
    .finally(() => setIsLoading(false));
}

return (
  <form onSubmit={handleSubmit}>
    <Textarea
      value={content}
      onChange={handleChange}
      placeholder="Be a part of the discussions"
      className={error ? 'border-primary border-2' : ''}
    />
    <small className={cn('invisible mt-1', error ? 'visible' : '')}>
      {error}
    </small>
    <Button type="submit" variant="destructive" className="my-8">
      <span className="mr-2">Add Comment</span>{' '}
      {isLoading && <LoadingSpinner size={18} />}
    </Button>
  </form>
);
```

```
}
```