SOLUTION - Answer Edit and Delete functionalty

1. Add the 2 functions needed in the question-actions.ts to Edit and Delete an answer.

```
export async function editAnswer(answerId: string, questionId: string, content: AnswerSchema)
{
    const result = await fetchClient(`/questions/${questionId}answers/${answerId}`,
        'PUT', {body: content});
    revalidatePath(`/questions/${questionId}`)
    return result;
}

export async function deleteAnswer(answerId: string, questionId: string) {
    const result = await fetchClient(`/questions/${questionId}/answers/${answerId}`,
        'DELETE');
    revalidatePath(`/questions/${questionId}`)
    return result;
}
```

2. Add the Edit and Delete buttons in the AnswerFooter. Position the buttons on the bottom left of the footer.

```
// AnswerFooter.tsx
    <div className='flex justify-between mt-4'>
            <div className='flex items-center mt-auto gap-1'>
                <>
                    <Button
                        size='sm'
                        variant='light'
                        color='primary'
                    >Edit</Button>
                    <Button
                        size='sm'
                        variant='light'
                        color='danger'
                    >Delete</Button>
                </>
            </div>

            <div className='flex flex-col basis-2/5 bg-primary/10 px-3 py-2 gap-2 rounded-
lg'>
                <span className='text-sm font-extralight'>answered
{timeAgo(answer.createdAt)}</span>
                <div className='flex items-center gap-3'>
                    <Avatar className='h-6 w-6' color='secondary'
                            name={answer.userDisplayName.charAt(0)} />
                    <div className='flex flex-col items-center'>
                        <span>{answer.userDisplayName}</span>
                        <span className='self-start text-sm font-semibold'>42</span>
                    </div>
                </div>
            </div>
        </div>
```

3. Ensure that only the user that answered the question can see these buttons.

```
export default async function AnswerContent({answer}: Props) {
    const currentUser = await getCurrentUser();
    return (
        <div className='flex border-b pb-3 px-6'>
```

```
            <VotingButtons accepted={answer.accepted} />
            <div className='flex flex-col w-full'>
                <div
                    className='flex-1 mt-4 ml-6 prose max-w-none dark:prose-invert'
                    dangerouslySetInnerHTML={{__html: answer.content}}
                />
                <AnswerFooter answer={answer} currentUser={currentUser} />
            </div>

        </div>
    );
}
```

```
// AnswerFooter.tsx

type Props = {
    answer: Answer;
    currentUser?: User | null;
}

export default function AnswerFooter({ answer, currentUser }: Props) {

// omitted code
```

```
            <div className='flex items-center mt-auto gap-1'>
                {currentUser?.id === answer.userId &&
                <>
                    <Button
                        isDisabled={!!editableAnswer}
                        onPress={() => {
                            setAnswer(answer);
                            setTimeout(() => {
                                document.getElementById('answer-form')?.scrollIntoView({
                                    behavior: 'smooth' });
                            }, 100);
                        }}
                        size='sm'
                        variant='light'
                        color='primary'
                    >Edit</Button>
                    <Button
                        isLoading={pending && answer.id === deleteTarget}
                        onPress={handleDelete}
                        size='sm'
                        variant='light'
                        color='danger'
                    >Delete</Button>
                </>}
            </div>
```

4. Add the function in the AnswerFooter.tsx to delete the answer. Ensure that the button displays a loading spinner when deleting.

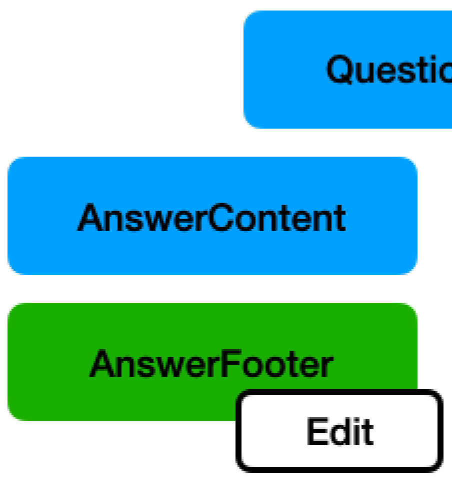
```
export default function AnswerFooter({ answer, currentUser }: Props) {
    const [pending, startTransition] = useTransition();
    const [deleteTarget, setDeleteTarget] = useState<string>('');

    const handleDelete = () => {
        setDeleteTarget(answer.id);
        startTransition(async () => {
            const {error} = await deleteAnswer(answer.id, answer.questionId);
            if (error) handleError(error);
            setDeleteTarget('');
        })
    }
```

```
    <Button
        isLoading={pending && answer.id === deleteTarget}
        onPress={handleDelete}
        size='sm'
        variant='light'
        color='danger'
>
        Delete
    </Button>
```

5. Test the delete answer functionality works.

Edit Functionality

This one is a bit trickier as we need to get the selected answer to edit from the user clicking on the edit button in the AnswerFooter into the AnswerForm.tsx. This diagram illustrates the challenge. The QuestionPage and AnswerContent are currently server components so we cannot use the useState hook to manage passing of state up to the QuestionsPage and down to the AnswerForm unless we turn everything in the diagram into a server component. Have we used anything else so far that can help us with this that would avoid the need to turn all these components into client components?



1. Ensure that when the edit button is clicked the following happens.
    1. The Answer is made available in the Answer form.
    2. The window scrolls down to the form ready for editing by the user. Since this is not a web design course the code to achieve that is provided here. You can give the outer div in the answer form an id property of 'answer-form' to make this work:

We need a zustand store for this to make it easy for ourselves. Create 'useAnswerStore' with the following code:

```
import { create } from 'zustand';
import {Answer} from "@/lib/types";

type AnswerStore = {
    answer: Answer | null;
    setAnswer: (answer: Answer) => void;
    clearAnswer: () => void;
};

export const useAnswerStore = create<AnswerStore>((set) => ({
    answer: null,
    setAnswer: (answer) => set({ answer }),
    clearAnswer: () => set({answer: null}),
}));
```

Update the AnswerFooter.tsx to use this store so that we can populate the answer when the Edit button is clicked.

```
export default function AnswerFooter({ answer, currentUser }: Props) {
    const [pending, startTransition] = useTransition();
    const [deleteTarget, setDeleteTarget] = useState<string>('');
    const setAnswer = useAnswerStore(state => state.setAnswer);
```

```
   return (
        <div className='flex justify-between mt-4'>
            <div className='flex items-center mt-auto gap-1'>
                {currentUser?.id === answer.userId &&
                <>
                    <Button
                        onPress={() => {
                            setAnswer(answer);
                            setTimeout(() => {
                                document.getElementById('answer-form')?.scrollIntoView({
                                    behavior: 'smooth' });
                            }, 100);
                        }}
                        size='sm'
                        variant='light'
                        color='primary'
                    >Edit</Button>
```

Give the outer div of the AnswerForm an Id so the scrollIntoView works

```
    return (
        <div id='answer-form' className='flex flex-col gap-3 items-start my-4 w-full px-6'>
            <h3 className='text-2xl'>Your answer</h3>
```

Use the useAnswerStore.ts in the AnswerForm.tsx so that we can get the answer.

```
type Props = {
    questionId: string;
}

export default function AnswerForm({questionId}: Props) {
    const [pending, startTransition] = useTransition();
    const editableAnswer = useAnswerStore(state => state.answer);
    const clearAnswer = useAnswerStore(state => state.clearAnswer);
```

We can use the values in the form to populate the content in the form with the answer (using a useEffect here and the reset would also work fine here):

```
    const {control, handleSubmit, reset, formState} = useForm<AnswerSchema>({
        mode: 'onTouched',
        resolver: zodResolver(answerSchema),
        values: {
            content: editableAnswer?.content
        }
    });
```

3. Add a cancel button to the form that will clear the form and the currently selected answer. Also update the text of the button to show edit or post depending on the existence of the selected answer in the form.

```
                <div className='flex items-start gap-3 mb-6'>
                    <Button
                        isDisabled={!formState.isValid || pending}
                        isLoading={pending}
                        color='primary'
                        className='w-fit'
                        type='submit'
                    >
                        {editableAnswer ? 'Update' : 'Post'} your answer
                    </Button>
                    <Button
                        isDisabled={!editableAnswer}
                        onPress={() => {
                            clearAnswer();
                            reset();
                        }}
                        className='w-fit'
                        type='button'
                    >
                        Cancel
                    </Button>
                </div>
```

4. Add the function into the AnswerForm.tsx to submit the edited form. Ensure the form and selected answer are cleared when this is submitted.

```
    const onSubmit = (data: AnswerSchema) => {
        startTransition(async () => {
            if (editableAnswer) {
                const {error} = await editAnswer(editableAnswer.id, 
editableAnswer.questionId, data);
                if (error) handleError(error);
                clearAnswer();
                reset();
            } else {
                const {error} = await postAnswer(data, questionId);
                if (error) handleError(error);
                reset();
            }
        })
    }
```

5. Also disable the edit button when an answer is selected for editing.

Test and finish.