```
import { AuthApiGuard } from '@/shared/utils/api/auth-guard';
import { supabaseAdmin } from '@/shared/utils/supabase/admin';
import { NextApiRequest, NextApiResponse } from 'next';
import { z } from 'zod';

//schema
const promptSchema = z.object({
  id: z.string(),
  name: z.string().min(2, { message: 'Name should be at least 2 characters' }),
  prompt: z
    .string()
    .min(5, { message: 'Prompt should be at least 5 characters' }),
  description: z.string().optional(),
  useCases: z
    .array(
      z.object({
        title: z.string().min(1, { message: 'Use case title is required' }),
        description: z
          .string()
          .min(1, { message: 'Use case description is required' }),
      }),
    )
    .min(1, { message: 'At least one use case is required' }),
  tags: z.string().optional(),
});
```

```
// Function to handle editing a prompt
const editPrompt = async (req: NextApiRequest, res: NextApiResponse) => {
  if (req.method !== 'POST') {
    res.setHeader('Allow', ['POST']);
    return res.status(405).end(`Method ${req.method} Not Allowed`);
  }

  try {
    const apiKey = req.headers.authorization?.split(' ')[1];
    if (!apiKey) {
      return res.status(401).json({ error: 'API Key is missing' });
    }

    const guard = new AuthApiGuard({ apiKey });
    const isAuthenticated = await guard.isAuthenticated();
    if (isAuthenticated.status !== 200) {
      return res
        .status(isAuthenticated.status)
        .json({ error: isAuthenticated.message });
    }

    const user_id = guard.getUserId();
    if (!user_id) {
      return res.status(404).json({ error: 'User is missing' });
    }
```

```
const input = promptSchema.parse(req.body);
const { id, name, prompt, description, useCases, tags } = input;

if (!id) {
  return res.status(404).json({
    error: 'Prompt ID not found',
  });
}

//check that prompt is for the authenticated user
const { data: existingPrompt, error: existingPromptError } =
  await supabaseAdmin
    .from('swarms_cloud_prompts')
    .select('*')
    .eq('user_id', user_id)
    .eq('id', id)
    .single();

if (existingPromptError) throw existingPromptError;
if (!existingPrompt) {
  return res.status(404).json({
    error: 'Prompt not found or you do not have permission to edit',
  });
}

//update the prompt
```

```
    const { data: updatedPrompt, error: updateError } = await supabaseAdmin
      .from('swarms_cloud_prompts')
      .update({
        name,
        use_cases: useCases,
        prompt,
        description,
        tags,
      })
      .eq('user_id', user_id)
      .eq('id', id)
      .select('*');

    if (updateError) throw updateError;

    return res.status(200).json(updatedPrompt);
  } catch (error) {
    console.error('An error occurred while editing prompt:', error);
    return res.status(500).json({ error: 'Internal Server Error' });
  }
};

export default editPrompt;
```