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

// Input validation schema
const editAgentSchema = z.object({
  id: z.string(),
  name: z.string().min(2, 'Name should be at least 2 characters'),
  agent: z
    .string()
    .min(5, { message: 'Agent should be at least 5 characters' }),
  language: z.string().optional(),
  description: z.string().min(1, 'Description is required'),
  requirements: z.array(
    z.object({
      package: z.string(),
      installation: z.string(),
    }),
  ),
  useCases: z.array(
    z.object({
      title: z.string(),
      description: z.string(),
    }),
  ),
```

```
  tags: z.string().optional(),
});

const editAgent = 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, go to link to create one',
        link: 'https://swarms.world/platform/api-keys',
      });
    }

    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 = editAgentSchema.parse(req.body);
const {
  id,
  name,
  agent,
  description,
  useCases,
  tags,
  language,
  requirements,
} = input;

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

const { data: existingAgent, error: existingAgentError } =
  await supabaseAdmin
    .from('swarms_cloud_agents')
```

```
    .select('*')
    .eq('user_id', user_id)
    .eq('id', id)
    .single();

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

//update agent
const { data: updatedAgent, error: updateError } = await supabaseAdmin
  .from('swarms_cloud_agents')
  .update({
    name,
    use_cases: useCases,
    agent,
    requirements,
    language,
    description,
    tags,
  })
  .eq('user_id', user_id)
  .eq('id', id)
```

```
      .select('*');

    if (updateError) throw updateError;

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

export default editAgent;
```