```
 }),
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 createAgentSchema = z.object({
  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().min(2, {
    message: 'Tags should be at least 1 characters and separated by commas',
  }),
});

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

// Rate limiting logic
const { data: lastSubmits, error: lastSubmitsError } = await supabaseAdmin
  .from('swarms_cloud_agents')
  .select('*')
  .eq('user_id', user_id)
  .order('created_at', { ascending: false })
  .limit(1);

if (lastSubmitsError) throw lastSubmitsError;

if (lastSubmits.length > 0) {
  const lastSubmit = lastSubmits[0];
  const lastSubmitTime = new Date(lastSubmit.created_at);
  const currentTime = new Date();
  const diffMinutes =
    (currentTime.getTime() - lastSubmitTime.getTime()) / (1000 * 60);
  if (diffMinutes < 1) {
    return res
      .status(429)
      .json({ error: 'You can only submit one agent per minute' });
  }
}

const { data: recentAgents, error: recentAgentsError } = await supabaseAdmin
  .from('swarms_cloud_agents')
  .select('*')
  .eq('agent', agent)
  .eq('user_id', user_id);

if (recentAgentsError) throw recentAgentsError;

if (recentAgents.length > 0) {
  return res.status(400).json({ error: 'Agent already exists' });
}

const trimTags = tags
  ?.split(',')
  .map((tag) => tag.trim())
  .filter(Boolean)
  .join(',');

const { error } = await supabaseAdmin.from('swarms_cloud_agents').insert([
      {
        name,
        use_cases: useCases,
        agent,
        description,
        user_id,
        requirements,
        language,
        tags: trimTags,
        status: 'pending',
      },
    ]);

    if (error) throw error;

    return res.status(200).json({ success: true });
  } catch (e) {
    console.error(e);
    if (e instanceof z.ZodError) {
      return res.status(400).json({ error: e.errors });
    }
    return res.status(500).json({ error: 'Could not add agent' });
  }
};

export default addAgent;
```