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

// Define a schema using Zod for validation
const TelemetryDataSchema = z.object({
  data: z.any(), // We only validate the data field as required
});

const logAgent = 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 authorizationHeader = req.headers.authorization;
    if (!authorizationHeader) {
      return res.status(401).json({
        error: 'Authorization header is missing',
        link: 'https://swarms.world/platform/api-keys',
      });
    }

    const apiKey = authorizationHeader.split(' ')[1];
```

```
if (!apiKey) {
  return res.status(401).json({
    error: 'API Key is missing in Authorization header',
    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 telemetryData = TelemetryDataSchema.parse(req.body);

const { data } = telemetryData;

const sourceIp =
  req.headers['x-forwarded-for'] || req.connection.remoteAddress || null;
```

```
  console.log('Source IP:', sourceIp);

  const { error } = await supabaseAdmin
    .from('swarms_framework_schema')
    .insert([
      {
        data: data || null,
        swarms_api_key: apiKey, // Use the API key from the Authorization header
      },
    ]);

  if (error) {
    throw new Error(`Supabase insert error: ${error.message}`);
  }

  console.log('Telemetry data successfully stored in Supabase');

  return res.status(200).json({
    message: 'Telemetry data received and stored successfully',
    data,
  });
} catch (error: any) {
  console.error('Telemetry API error', error);
  return res
    .status(500)
    .json({ error: 'Internal Server Error', details: error.message });
```

```
  }
};


export default logAgent;
```