```
import { Tables } from '@/types_db';
import { supabaseAdmin } from '../supabase/admin';
import { getUserOrganizationRole } from './organization';
import { checkRateLimit } from './rate-limit';
import { getBillingLimit } from './usage';
import { currentMonth } from '@/shared/constants/date';

type Options = {
  apiKey: string | null;
  organizationPublicId: string | null;
  modelId: string | null;
};
type UsageOptions = {
  model: string;
  temperature: number;
  top_p: number;
  echo?: boolean;
  stream?: boolean;
  input_cost: number;
  output_cost: number;
  total_cost: number;
  input_tokens: number;
  output_tokens: number;
  max_tokens: number;
  messages: any;
  invoice_total_cost?: number;
```

```
};
export class SwarmsApiGuard {
  apiKey: string | null;
  organizationPublicId: string | null;
  organizationId: string | null = null;
  modelId: string | null;
  modelRecord: Tables<'swarms_cloud_models'> | null = null;
  apiKeyRecordId: string | null = null;
  userId: string | null = null;

  constructor({ apiKey, organizationPublicId, modelId }: Options) {
    this.apiKey = apiKey;
    this.organizationPublicId = organizationPublicId;
    this.modelId = modelId;
  }
  async isAuthenticated(): Promise<{
    status: number;
    message: string;
  }> {
    if (!this.apiKey) {
      return { status: 401, message: 'API Key is missing' };
    }

    // api key validation
    const apiKeyInfo = await supabaseAdmin
      .from('swarms_cloud_api_keys')
```

```
  .select('*')
  .eq('key', this.apiKey)
  .neq('is_deleted', true)
  .maybeSingle();

if (!apiKeyInfo.data?.id) {
  return { status: 401, message: 'Invalid API Key' };
}
this.apiKeyRecordId = apiKeyInfo.data.id;
this.userId = apiKeyInfo.data.user_id;

if (!this.modelId) {
  return { status: 400, message: 'model is missing' };
}

// check organization validation if provided
if (this.organizationPublicId) {
  const org = await supabaseAdmin
    .from('swarms_cloud_organizations')
    .select('*')
    .eq('public_id', this.organizationPublicId)
    .maybeSingle();
  if (!org.data?.id) {
    return { status: 404, message: 'Organization not found' };
  }
  const orgId = org.data?.id;
```

```
this.organizationId = orgId;
const userRoleInOrg = await getUserOrganizationRole(orgId, this.userId);
if (!userRoleInOrg) {
  return { status: 401, message: 'User is not part of the organization' };
}

// check rate limit - set for organizations only
const orgMembers = await supabaseAdmin
  .from('swarms_cloud_organization_members')
  .select('user_id')
  .eq('organization_id', orgId);

const onlyOrgMembers = (orgMembers.data || [])?.filter(
  (m) => m.user_id !== org.data?.owner_user_id,
);

for (const member of onlyOrgMembers) {
  const isAllowed = await checkRateLimit(member.user_id ?? '', 50);
  if (!isAllowed) {
    return {
      status: 429,
      message: 'Too Many Requests. Please try again later.',
    };
  }
}
```

```
  const isAllowed = await checkRateLimit(org.data.owner_user_id ?? '', 100);
  if (!isAllowed) {
    return {
      status: 429,
      message: 'Too Many Requests. Please try again later.',
    };
  }
}

const isBillingAllowed = await getBillingLimit(this.userId, currentMonth);
if (isBillingAllowed.status !== 200) {
  return {
    status: isBillingAllowed.status,
    message: isBillingAllowed.message,
  };
}

// check user is not banned: SOON

// check rate limit - set for users only
if (!this.organizationPublicId) {
  const isAllowed = await checkRateLimit(this.userId, 50);
  if (!isAllowed) {
    return {
      status: 429,
      message: 'Too Many Requests. Please try again later.',
```

```
      };
    }
  }

  // check model exists
  const modelInfo = await supabaseAdmin
    .from('swarms_cloud_models')
    .select('*')
    .eq('unique_name', this.modelId)
    .neq('enabled', false)
    .maybeSingle();

  if (!modelInfo?.data?.id) {
    return { status: 404, message: 'Model not found' };
  }
  this.modelRecord = modelInfo.data;

  return { status: 200, message: 'Success' };
}

getUserId(): string | null {
  return this.userId;
}

async logUsage(usage: UsageOptions): Promise<{
  status: number;
```

```
  message: string;
}> {
  if (!this.modelRecord) {
    return { status: 400, message: 'Model is missing' };
  }
  if (!this.userId) {
    return { status: 400, message: 'User is missing' };
  }
  if (!this.apiKeyRecordId) {
    return { status: 400, message: 'API Key is missing' };
  }

  const activity = await supabaseAdmin
    .from('swarms_cloud_api_activities')
    .insert([
      {
        api_key_id: this.apiKeyRecordId,
        organization_id: this.organizationId,
        user_id: this.userId,
        model_id: this.modelRecord.id,
        repetition_penalty: 1,
        temperature: usage.temperature,
        top_p: usage.top_p,
        echo: usage.echo,
        stream: usage.stream,
        input_cost: usage.input_cost,
```

```
        output_cost: usage.output_cost,
        total_cost: usage.total_cost,
        input_tokens: usage.input_tokens,
        output_tokens: usage.output_tokens,
        max_tokens: usage.max_tokens,
        messages: usage.messages,
        invoice_total_cost: usage.invoice_total_cost,
      } as unknown as Tables<'swarms_cloud_api_activities'>,
    ]);

  if (activity.error) {
    console.log('error', activity.error);

    return {
      status: 500,
      message: 'Internal Server Error - Swarms activity log failed',
    };
  }

  return {
    status: 200,
    message: 'Success',
  };
 }
}
```