```
import { NextApiRequest, NextApiResponse } from 'next';
import { BillingService } from '@/shared/utils/api/billing-service';
import { supabaseAdmin } from '@/shared/utils/supabase/admin';
import { chunk } from '@/shared/utils/helpers';
import { User } from '@supabase/supabase-js';

const BATCH_SIZE = 50;

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse,
) {
  try {
    const authorizationHeader = req.headers?.['Authorization'];

    if (
      !authorizationHeader ||
      authorizationHeader !== `Bearer ${process.env.CRON_SECRET}`
    ) {
      return res.status(401).end('Unauthorized');
    }

    const currentDate = new Date();
    const lastMonthDate = new Date(
      currentDate.getFullYear(),
      currentDate.getMonth() - 1,
```

```
  1,
  1,
);

if (currentDate.getDate() === lastMonthDate.getDate()) {
  console.log('Skipping invoice generation for current month');
  return res
    .status(200)
    .json({ message: 'Skipping invoice generation for current month' });
}

const { data: allUsers, error: fetchError } = await supabaseAdmin
  .from('users')
  .select('*');

if (fetchError) {
  console.error('Error fetching users:', fetchError);
  return res.status(500).json({ message: 'Internal server error' });
}

if (!allUsers || allUsers.length === 0) {
  console.log('No users found');
  return res.status(404).json({ message: 'No users found' });
}

const userBatches = chunk(allUsers, BATCH_SIZE);
```

```
await Promise.all(
  userBatches.map(async (batch) => {
    await Promise.all(
      batch.map(async (user) => {
        const billingService = new BillingService(user.id);
        const usage =
          await billingService.calculateTotalMonthlyUsage(lastMonthDate);

        if (usage.status !== 200) {
          console.error(
            'Error calculating total monthly usage:',
            usage.message,
          );
          return res.status(500).json({ message: 'Internal server error' });
        }

        let totalAmount = usage.user.totalCost;
        let invoiceDescription = `Monthly API Usage billing ${user.email}`;

        usage.organizations.forEach((org) => {
          if (org.ownerId === user.id) {
            totalAmount += org.totalCost;
            invoiceDescription = `Monthly API Usage billing for ${org.name}`;
          }
        });
```

```
            await billingService.sendInvoiceToUser(
              totalAmount,
              user as unknown as User,
              invoiceDescription,
            );
          }),
        );
      }),
    );

    return res.status(200).json({ message: 'Invoice generation successful' });
  } catch (error) {
    console.error('Error sending invoices:', error);
    return res.status(500).send('Something definitely went wrong');
  }
}
```