```
import type { Tables } from '@/types_db';
import confetti from 'canvas-confetti';

type Price = Tables<'prices'>;

interface ObjectOrArray {
  constructor: typeof Object | typeof Array;
}

export type ShareDetails = {
  message: string;
  link: string;
  subject?: string;
};

export const getURL = (path: string = '') => {
  // Check if NEXT_PUBLIC_SITE_URL is set and non-empty. Set this to your site URL in production
env.
  let url =
    process?.env?.NEXT_PUBLIC_SITE_URL &&
    process.env.NEXT_PUBLIC_SITE_URL.trim() !== ''
      ? process.env.NEXT_PUBLIC_SITE_URL
      : // If not set, check for NEXT_PUBLIC_VERCEL_URL, which is automatically set by Vercel.
        process?.env?.NEXT_PUBLIC_VERCEL_URL &&
          process.env.NEXT_PUBLIC_VERCEL_URL.trim() !== ''
        ? process.env.NEXT_PUBLIC_VERCEL_URL
        : // If neither is set, default to localhost for local development.
          'http://localhost:3000/';

  // Trim the URL and remove trailing slash if exists.
  url = url.replace(/\/+$/, '');
  // Make sure to include `https://` when not localhost.
  url = url.includes('http') ? url : `https://${url}`;
  // Ensure path starts without a slash to avoid double slashes in the final URL.
  path = path.replace(/^\/+/, '');

  // Concatenate the URL and the path.
  const fullPath = `${url}/${path}`;
  return fullPath;
};

export const postData = async ({
  url,
  data,
}: {
  url: string;
  data?: { price: Price };
}) => {
  const res = await fetch(url, {
    method: 'POST',
    headers: new Headers({ 'Content-Type': 'application/json' }),
    credentials: 'same-origin',
    body: JSON.stringify(data),
  });

  return res.json();
};

export const toDateTime = (secs: number) => {
  var t = new Date(+0); // Unix epoch start.
  t.setSeconds(secs);
  return t;
};

export const calculateTrialEndUnixTimestamp = (
  trialPeriodDays: number | null | undefined,
) => {
  // Check if trialPeriodDays is null, undefined, or less than 2 days
  if (
    trialPeriodDays === null ||
    trialPeriodDays === undefined ||
    trialPeriodDays < 2
  ) {
    return undefined;
  }

  const currentDate = new Date(); // Current date and time
  const trialEnd = new Date(
    currentDate.getTime() + (trialPeriodDays + 1) * 24 * 60 * 60 * 1000,
  ); // Add trial days
  return Math.floor(trialEnd.getTime() / 1000); // Convert to Unix timestamp in seconds
};

const toastKeyMap: { [key: string]: string[] } = {
  status: ['status', 'status_description'],
  error: ['error', 'error_description'],
};

const getToastRedirect = (
  path: string,
  toastType: string,
  toastName: string,
  toastDescription: string = '',
  disableButton: boolean = false,
  arbitraryParams: string = '',
): string => {
  const [nameKey, descriptionKey] = toastKeyMap[toastType];

  let redirectPath = `${path}?${nameKey}=${encodeURIComponent(toastName)}`;

  if (toastDescription) {
    redirectPath += `&${descriptionKey}=${encodeURIComponent(toastDescription)}`;
  }
  if (disableButton) {
    redirectPath += `&disable_button=true`;
  }

  if (arbitraryParams) {
    redirectPath += `&${arbitraryParams}`;
  }

  return redirectPath;
};

export const getStatusRedirect = (
  path: string,
  statusName: string,
  statusDescription: string = '',
  disableButton: boolean = false,
  arbitraryParams: string = '',
) =>
  getToastRedirect(
    path,
    'status',
    statusName,
    statusDescription,
    disableButton,
    arbitraryParams,
  );
```

```
export const getErrorRedirect = (
  path: string,
  errorName: string,
  errorDescription: string = '',
  disableButton: boolean = false,
  arbitraryParams: string = '',
) =>
  getToastRedirect(
    path,
    'error',
    errorName,
    errorDescription,
    disableButton,
    arbitraryParams,
  );

export function generateApiKey() {
  // Generate a random hexadecimal string of length 64
  var randomHex = '';
  for (var i = 0; i < 64; i++) {
    randomHex += Math.floor(Math.random() * 16).toString(16);
  }

  // Construct the API key in the required format
  var apiKey = 'sk-' + randomHex;
  return apiKey;
}


export const formatDate = (date: string) => {
  // like: Jul 28, 2022
  return new Date(date).toLocaleDateString('en-US', {
    year: 'numeric',
    month: 'short',
    day: 'numeric',
  });
};


export const commaSeparated = (value: number) =>
  value.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ',');


export const formatSpentTime = (value: number) => {
  // convert seconds to : months, days, hours, minutes, seconds
  const months = Math.floor(value / 2592000);
  const days = Math.floor((value % 2592000) / 86400);
  const hours = Math.floor((value % 86400) / 3600);
  const minutes = Math.floor((value % 3600) / 60);
  const seconds = value % 60;


  // return the biggest unit that is not 0
  if (months > 0) {
    return `${months} month${months > 1 ? 's' : ''}`;
  } else if (days > 0) {
    return `${days} day${days > 1 ? 's' : ''}`;
  } else if (hours > 0) {
    return `${hours} hour${hours > 1 ? 's' : ''}`;
  } else if (minutes > 0) {
    return `${minutes} minute${minutes > 1 ? 's' : ''}`;
  } else {
    return `${seconds} second${seconds > 1 ? 's' : ''}`;
  }
};

export const makeUrl = (url: string, data: any) => {
  // replace all [key], {key} with data[key]
  return url.replace(/\[(.*?)\]|\{(.*?)\}/g, (match, p1, p2) => {
    return data[p1 || p2];
  });
};

// shorten string to num and attached endLabel to shortened string
export function getTruncatedString(str: string, num: number, endLabel = '...') {
  if (!str) return null;

  const words = str.split('').splice(0, num);
  if (str.split('').length > num) return `${words.join('')}${endLabel}`;
  return str;
}
```

```
export const isEmpty = (obj: ObjectOrArray | any) =>
  [Object, Array].includes((obj || {}).constructor) &&
  !Object.entries(obj || {}).length;

export const debounce = (callback: (...args: any[]) => any, wait: number) => {
  let timeoutId: number | undefined;
  return (...args: any[]) => {
    window.clearTimeout(timeoutId);
    timeoutId = window.setTimeout(() => {
      callback(...args);
    }, wait);
  };
};

export function throttle<T extends (...args: any[]) => any>(
  func: T,
  limit: number,
): (...args: Parameters<T>) => void {
  let inThrottle: boolean;

  return function (this: any, ...args: Parameters<T>) {
    const context = this;
    if (!inThrottle) {
      func.apply(context, args);
      inThrottle = true;
      setTimeout(() => (inThrottle = false), limit);
    }
  };
}

export function formatPrice(
  price: number | string,
  currencyCode = 'USD',
  locale = 'en-US',
): string {
  if (!price) return '';

  const hasDecimalPlaces = price.toString().includes('.');

  return new Intl.NumberFormat(locale, {
    style: 'currency',
    currency: currencyCode,
    minimumFractionDigits: hasDecimalPlaces ? 2 : 0,
  }).format(Number(price));
}

/**
 * _.chunk(['a', 'b', 'c', 'd'], 2);
 * => [['a', 'b'], ['c', 'd']]
 * @returns an array of elements split into groups the length of size
 */
```

```
export const chunk = <T>(input: T[], size: number): T[][] => {
  return input.reduce<T[][]>((arr, item, idx) => {
    return idx % size === 0
      ? [...arr, [item]]
      : [...arr.slice(0, -1), [...arr.slice(-1)[0], item]];
  }, []);
};


export const createQueryString = (params: Record<string, string>) => {
  const searchParams = new URLSearchParams();


  Object.entries(params).forEach(([name, value]) => {
    searchParams.set(name, value);
  });


  return searchParams.toString();
};


export const openShareWindow = (platform: string, details: ShareDetails) => {
  const { message, link, subject } = details;
  const encodedLink = encodeURIComponent(link);
  let url = '';


  switch (platform) {
    case 'twitter':
                                                                                                                                                    url                    =
`https://twitter.com/intent/tweet?text=${encodeURIComponent(message)}&url=${encodedLink}`;
      break;
    case 'linkedin':
      url = `https://www.linkedin.com/sharing/share-offsite/?url=${encodedLink}`;
      break;
    case 'facebook':
      url = `https://www.facebook.com/sharer/sharer.php?u=${encodedLink}`;
      break;
    case 'reddit':
                                                                      url                =
`https://www.reddit.com/submit?url=${encodedLink}&title=${encodeURIComponent(message)}`;
      break;
    case 'hackernews':
                                                                      url                =
`https://news.ycombinator.com/submitlink?u=${encodedLink}&t=${encodeURIComponent(message)
}`;
      break;
    case 'email':
      if (subject) {
        const encodedSubject = encodeURIComponent(subject);
        const encodedMessage = encodeURIComponent(`${message}\n\n${link}`);
                                                                        url            =
`https://mail.google.com/mail/?view=cm&fs=1&to=&su=${encodedSubject}&body=${encodedMessa
ge}`;
      }
      break;
    default:
      console.error('Unsupported platform');
      return;
  }

  window.open(url, '_blank');
};

export const getMonthStartEndDates = (month: Date) => {
  const start = new Date(
    month.getFullYear(),
    month.getMonth(),
    1,
    1,
  ).toISOString();

  const end = new Date(
    month.getFullYear(),
    month.getMonth() + 1,
    0,
    24,
    59,
    59,
    999,
  ).toISOString();
  return { start, end };
};

// Function to launch confetti
export const launchConfetti = () => {
  const duration = 4000;
  const interval = 1000;
  const colors = [
    '#ff0000',
    '#ff7f00',
    '#ffff00',
    '#00ff00',
    '#0000ff',
    '#4b0082',
    '#8b00ff',
  ]; // rainbow colors

  // Function to get a random number within a range
  const randomInRange = (min: number, max: number) =>
    Math.random() * (max - min) + min;

  const animationEnd = Date.now() + duration;

  const fireConfetti = () => {
    if (Date.now() < animationEnd) {
      const count = 3; // Number of firework bursts
      for (let i = 0; i < count; i++) {
        const x = Math.random() * window.innerWidth;
        const y = Math.random() * window.innerHeight;
        confetti({
          particleCount: 50, // Low particle count for a subtle effect
          angle: randomInRange(55, 125), // Random angle for fireworks effect
          spread: 80,
          startVelocity: randomInRange(45, 65),
          decay: 0.95,
          colors: colors,
          origin: { x: x / window.innerWidth, y: y / window.innerHeight },
        });
      }
      setTimeout(fireConfetti, interval);
    }
  };

  fireConfetti();
};
```