File: client/src/lib/adManager.ts

Type: ts | **Size:** 4820 bytes

```
// Ad Manager for Love Oracle App
// This utility will handle ad display logic and tracking
import { PreferenceKeys, getBooleanValue, setBooleanValue, getNumberValue, setNumberValue } from './preferen
ce...
// Constants for ad configuration
export const AD_CONFIG = {
  REMOVE_ADS_PRICE: 4.99,
  INTERSTITIAL_FREQUENCY: 2, // Show interstitial after every 2 actions
  MIN_INTERSTITIAL_INTERVAL: 120000, // Minimum time between interstitials in ms (2 minutes)
};
// Storage keys for preferences
const STORAGE_KEYS = {
  AD_FREE: PreferenceKeys.AD_FREE,
  ACTION_COUNT: PreferenceKeys.ACTION_COUNT,
  LAST_AD_SHOWN: PreferenceKeys.LAST_AD_SHOWN,
};
// Classes to identify ad containers
export const AD_CLASSES = {
  BANNER: 'admob-banner-container',
  INTERSTITIAL: 'admob-interstitial-trigger',
};
// Real ad unit IDs from Google AdMob
export const AD_UNIT_IDS = {
  BANNER: 'ca-app-pub-5717347186318471/7831347387',
  INTERSTITIAL: 'ca-app-pub-3940256099942544/1033173712', // Test ID - Replace with actual
};
// Store for in-memory values since Preferences is async
let _cachedAdFree: boolean | null = null;
let _cachedActionCount: number | null = null;
/**
 * Check if user has ad-free experience
 */
export function isAdFree(): boolean {
  // Use cached value if available
  if (_cachedAdFree !== null) return _cachedAdFree;

  // Fallback to localStorage until async value is retrieved
  return localStorage.getItem('loveoracle_ad_free') === 'true';
}
/**
 * Set ad-free status
 */
export function setAdFree(value: boolean): void {
  // Update cache
  _cachedAdFree = value;

  // Store in Preferences (async)
  setBooleanValue(STORAGE_KEYS.AD_FREE, value);

  // Also set in localStorage for immediate fallback
  localStorage.setItem('loveoracle_ad_free', value ? 'true' : 'false');
}
/**
 * Get current action count for interstitial ads
 */
export function getActionCount(): number {
  // Use cached value if available
  if (_cachedActionCount !== null) return _cachedActionCount;

  // Fallback to localStorage until async value is retrieved
  const count = localStorage.getItem('loveoracle_action_count');
  return count ? parseInt(count, 10) : 0;
}
/**
 * Increment action count and check if interstitial should be shown
 * Returns true if an interstitial should be shown
 */
export function incrementActionCount(): boolean {
  if (isAdFree()) return false;

  const currentCount = getActionCount();
  const newCount = currentCount + 1;
  // Update cached value
  _cachedActionCount = newCount;

  // Store in Preferences (async)
  setNumberValue(STORAGE_KEYS.ACTION_COUNT, newCount);

  // Also set in localStorage for immediate fallback
  localStorage.setItem('loveoracle_action_count', newCount.toString());

  // Show interstitial ad every N actions
  if (newCount % AD_CONFIG.INTERSTITIAL_FREQUENCY === 0) {
    const now = Date.now();

    // Store last interstitial time
    setNumberValue(STORAGE_KEYS.LAST_AD_SHOWN, now);
    localStorage.setItem('loveoracle_last_interstitial_time', now.toString());

    return true;
  }

  return false;
}
/**
 * Reset action count
 */
export function resetActionCount(): void {
  // Update cached value
  _cachedActionCount = 0;

  // Store in Preferences (async)
  setNumberValue(STORAGE_KEYS.ACTION_COUNT, 0);

  // Also set in localStorage
  localStorage.setItem('loveoracle_action_count', '0');
}
/**
 * Check if enough time has passed since last interstitial
 * to prevent too frequent interstitials
 */
export function canShowInterstitial(): boolean {
  if (isAdFree()) return false;

  // Use localStorage as immediate fallback
  const lastTimeStr = localStorage.getItem('loveoracle_last_interstitial_time');
  if (!lastTimeStr) return true;

  const lastTime = parseInt(lastTimeStr, 10);
  const now = Date.now();

  // Minimum interval between interstitials
  return (now - lastTime) > AD_CONFIG.MIN_INTERSTITIAL_INTERVAL;
}
/**
 * Initialize ad-related data if not already set
 */
export function initAdManager(): void {
  // Initial async load of values from Preferences to cache
  getBooleanValue(STORAGE_KEYS.AD_FREE, false).then(value => {
    _cachedAdFree = value;
    // Also update localStorage
    localStorage.setItem('loveoracle_ad_free', value ? 'true' : 'false');
  });

  getNumberValue(STORAGE_KEYS.ACTION_COUNT, 0).then(value => {
    _cachedActionCount = value;
    // Also update localStorage
    localStorage.setItem('loveoracle_action_count', value.toString());
  });

  // Set default values if not in localStorage yet
  if (localStorage.getItem('loveoracle_ad_free') === null) {
    localStorage.setItem('loveoracle_ad_free', 'false');
  }

  if (localStorage.getItem('loveoracle_action_count') === null) {
    localStorage.setItem('loveoracle_action_count', '0');
    }
}
```