File: client/src/lib/adMobService.ts

Type: ts | **Size:** 6297 bytes

```
import { AdMob, BannerAdPluginEvents, BannerAdSize, BannerAdPosition, InterstitialAdPluginEvents } from '@ca
pa...
import { Device } from '@capacitor/device';
import { getActionCount, incrementActionCount, resetActionCount, canShowInterstitial } from './adManager';
// Define banner ad positions
export enum AdPosition {
  TOP = 'top',
  BOTTOM = 'bottom'
}
// Helper function to check platform
export async function checkPlatform(platform: 'android' | 'ios' | 'web'): Promise {
  try {
    const info = await Device.getInfo();
    return info.platform === platform;
  } catch (e) {
    console.error('Error checking platform:', e);
    return false;
  }
}
// Ad Unit IDs - replace test IDs with actual IDs when ready for production
const BANNER_AD_ID = 'ca-app-pub-5717347186318471/7831347387';
// Using a test ID for interstitial since the actual ID wasn't provided
const INTERSTITIAL_AD_ID = 'ca-app-pub-3940256099942544/1033173712'; // Test ID - Replace with actual
// Initialize AdMob
export async function initializeAdMob(): Promise {
  try {
    // Check if we're on a native platform
    const isAndroid = await checkPlatform('android');
    const isIOS = await checkPlatform('ios');

    if (isAndroid || isIOS) {
      // Request tracking authorization (primarily for iOS)
      await AdMob.requestTrackingAuthorization();

      // Initialize AdMob with your app ID
      const initialize = await AdMob.initialize({
        // These are the correct options for AdMob initialization
        tagForChildDirectedTreatment: true,
        // Removed npa option as it's not in the type definition
      });

      console.log('AdMob initialized', initialize);
      return true;
    } else {
      console.log('Not running on a native platform, AdMob not initialized');
      return false;
    }
  } catch (error) {
    console.error('Error initializing AdMob:', error);
    return false;
  }
}
// Show a banner ad at the specified position
export async function showBannerAd(position: AdPosition = AdPosition.TOP): Promise {
  try {
    // Check if we're on a native platform
    const isAndroid = await checkPlatform('android');
    const isIOS = await checkPlatform('ios');

    if (!isAndroid && !isIOS) {
      console.log('Not running on a native platform, banner ad not shown');
      return false;
    }

    // Remove any existing banner first
    await AdMob.removeBanner();

    // Add listeners for banner events
    AdMob.addListener(BannerAdPluginEvents.Loaded, () => {
      console.log('Banner ad loaded');
    });

    AdMob.addListener(BannerAdPluginEvents.SizeChanged, (size) => {
      console.log('Banner size changed:', size);
    });

    AdMob.addListener(BannerAdPluginEvents.FailedToLoad, (error) => {
      console.error('Failed to load banner ad:', error);
    });

    // Show the banner
    await AdMob.showBanner({
      adId: BANNER_AD_ID,
      adSize: BannerAdSize.ADAPTIVE_BANNER,
      // Use the proper enum values from the AdMob library
      position: position === AdPosition.TOP ? BannerAdPosition.TOP_CENTER : BannerAdPosition.BOTTOM_CENTER,
      margin: 0,
    });

    return true;
  } catch (error) {
    console.error('Error showing banner ad:', error);
    return false;
  }
}
// Hide the banner ad
export async function hideBannerAd(): Promise {
  try {
    // Check if we're on a native platform
    const isAndroid = await checkPlatform('android');
    const isIOS = await checkPlatform('ios');

    if (isAndroid || isIOS) {
      await AdMob.hideBanner();
      return true;
    }
    return false;
  } catch (error) {
    console.error('Error hiding banner ad:', error);
    return false;
  }
}
// Prepare an interstitial ad
let interstitialPrepared = false;
export async function prepareInterstitial(): Promise {
  try {
    // Check if we're on a native platform
    const isAndroid = await checkPlatform('android');
    const isIOS = await checkPlatform('ios');

    if (!isAndroid && !isIOS) {
      return false;
    }

    // Add listeners for interstitial events
    AdMob.addListener(InterstitialAdPluginEvents.Loaded, () => {
      console.log('Interstitial ad loaded');
      interstitialPrepared = true;
    });

    AdMob.addListener(InterstitialAdPluginEvents.FailedToLoad, (error) => {
      console.error('Failed to load interstitial ad:', error);
      interstitialPrepared = false;
    });

    AdMob.addListener(InterstitialAdPluginEvents.Dismissed, () => {
      console.log('Interstitial ad dismissed');
      interstitialPrepared = false;
      // Prepare the next interstitial
      prepareInterstitial();
    });

    // Prepare the interstitial
    await AdMob.prepareInterstitial({
      adId: INTERSTITIAL_AD_ID
    });

    return true;
  } catch (error) {
    console.error('Error preparing interstitial ad:', error);
    interstitialPrepared = false;
    return false;
  }
}
// Show an interstitial ad if conditions are met
export async function showInterstitialIfReady(): Promise {
  try {
    // Check if we're on a native platform
    const isAndroid = await checkPlatform('android');
    const isIOS = await checkPlatform('ios');

    if (!isAndroid && !isIOS) {
      return false;
    }

    // Check if action count is high enough and enough time has passed
    if (incrementActionCount() && canShowInterstitial() && interstitialPrepared) {
      // Show the interstitial
      await AdMob.showInterstitial();
      // Reset the action count after showing
      resetActionCount();
      return true;
    }

    return false;
  } catch (error) {
    console.error('Error showing interstitial ad:', error);
    return false;
  }
}
// Track user action that may trigger an interstitial
export function trackAction(): boolean {
  // Just increment the action count, but don't show an ad immediately
  // This allows the ad to be shown at an appropriate time
  const shouldShowAd = incrementActionCount() && canShowInterstitial();

  // If conditions are met, prepare to show an ad on next appropriate moment
  if (shouldShowAd && !interstitialPrepared) {
    prepareInterstitial();
  }

  return shouldShowAd && interstitialPrepared;
}
```