File: client/src/lib/preferencesService.ts

Type: ts | Size: 7026 bytes

```
import { Preferences } from '@capacitor/preferences';
import { useCapacitor } from '@/hooks/use-capacitor';

/**
 * Keys used for storing values in the Preferences storage
 */
export enum PreferenceKeys {
  USER_ID = 'user_id',
  USERNAME = 'username',
  AD_FREE = 'ad_free',
  SUBSCRIPTION_STATUS = 'subscription_status',
  SUBSCRIPTION_EXPIRY = 'subscription_expiry',
  LAST_AD_SHOWN = 'last_ad_shown',
  ACTION_COUNT = 'action_count',
  DAILY_CARD_DATE = 'daily_card_date',
  THEME = 'theme',
  NOTIFICATION_ENABLED = 'notification_enabled',
  ONBOARDING_COMPLETED = 'onboarding_completed',
}

/**
 * Save a string value in storage
 */
export async function setStringValue(key: PreferenceKeys | string, value: string): Promise<void> {
  try {
    await Preferences.set({
      key,
      value,
    });
  } catch (error) {
    console.error(`Error saving preference ${key}:`, error);
    // Fall back to localStorage for web
    try {
      localStorage.setItem(key, value);
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
    }
  }
}

/**
 * Save a boolean value in storage
 */
export async function setBooleanValue(key: PreferenceKeys | string, value: boolean): Promise<void> {
  try {
    await Preferences.set({
      key,
      value: value.toString(),
    });
  } catch (error) {
    console.error(`Error saving preference ${key}:`, error);
    // Fall back to localStorage for web
    try {
      localStorage.setItem(key, value.toString());
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
    }
  }
}

/**
 * Save a numeric value in storage
 */
export async function setNumberValue(key: PreferenceKeys | string, value: number): Promise<void> {
  try {
    await Preferences.set({
      key,
      value: value.toString(),
    });
  } catch (error) {
    console.error(`Error saving preference ${key}:`, error);
    // Fall back to localStorage for web
    try {
      localStorage.setItem(key, value.toString());
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
    }
  }
}

/**
 * Save an object value in storage (serialized as JSON)
 */
export async function setObjectValue(key: PreferenceKeys | string, value: object): Promise<void> {
  try {
    await Preferences.set({
      key,
      value: JSON.stringify(value),
    });
  } catch (error) {
    console.error(`Error saving preference ${key}:`, error);
    // Fall back to localStorage for web
    try {
      localStorage.setItem(key, JSON.stringify(value));
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
    }
  }
}

/**
 * Get a string value from storage
 */
export async function getStringValue(key: PreferenceKeys | string, defaultValue: string = ''): Promise<string>...
  try {
    const { value } = await Preferences.get({ key });
    return value || defaultValue;
  } catch (error) {
    console.error(`Error retrieving preference ${key}:`, error);
    // Fall back to localStorage for web
    try {
      const value = localStorage.getItem(key);
      return value || defaultValue;
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
      return defaultValue;
    }
  }
}

/**
 * Get a boolean value from storage
 */
export async function getBooleanValue(key: PreferenceKeys | string, defaultValue: boolean = false): Promise<bo...
  try {
    const { value } = await Preferences.get({ key });
    return value === 'true' ? true : value === 'false' ? false : defaultValue;
  } catch (error) {
    console.error(`Error retrieving preference ${key}:`, error);
    // Fall back to localStorage for web
    try {
      const value = localStorage.getItem(key);
      return value === 'true' ? true : value === 'false' ? false : defaultValue;
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
      return defaultValue;
    }
  }
}

/**
 * Get a numeric value from storage
 */
```

```
export async function getNumberValue(key: PreferenceKeys | string, defaultValue: number = 0): Promise<number> ...
  try {
    const { value } = await Preferences.get({ key });
    return value ? Number(value) : defaultValue;
  } catch (error) {
    console.error(`Error retrieving preference ${key}:`, error);
    // Fall back to localStorage for web
    try {
      const value = localStorage.getItem(key);
      return value ? Number(value) : defaultValue;
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
      return defaultValue;
    }
  }
}

/**
 * Get an object value from storage (parsed from JSON)
 */
export async function getObjectValue<T>(key: PreferenceKeys | string, defaultValue: T): Promise<T> {
  try {
    const { value } = await Preferences.get({ key });
    return value ? JSON.parse(value) : defaultValue;
  } catch (error) {
    console.error(`Error retrieving preference ${key}:`, error);
    // Fall back to localStorage for web
    try {
      const value = localStorage.getItem(key);
      return value ? JSON.parse(value) : defaultValue;
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
      return defaultValue;
    }
  }
}

/**
 * Remove a value from storage
 */
export async function removeValue(key: PreferenceKeys | string): Promise<void> {
  try {
    await Preferences.remove({ key });
  } catch (error) {
    console.error(`Error removing preference ${key}:`, error);
    // Fall back to localStorage for web
    try {
      localStorage.removeItem(key);
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
    }
  }
}

/**
 * Clear all values from storage
 */
export async function clearAll(): Promise<void> {
  try {
    await Preferences.clear();
  } catch (error) {
    console.error('Error clearing preferences:', error);
    // Fall back to localStorage for web
    try {
      localStorage.clear();
    } catch (e) {
      console.error('Failed to fallback to localStorage for clearing:', e);
    }
  }
}

/**
 * Check if a key exists in storage
 */
export async function keyExists(key: PreferenceKeys | string): Promise<boolean> {
  try {
    const { value } = await Preferences.get({ key });
    return value !== null && value !== undefined;
  } catch (error) {
    console.error(`Error checking if key ${key} exists:`, error);
    // Fall back to localStorage for web
    try {
      return localStorage.getItem(key) !== null;
    } catch (e) {
      console.error(`Failed to fallback to localStorage for ${key}:`, e);
      return false;
    }
  }
}

// Hook for accessing Preferences in components
export function usePreferences() {
  const { isCapacitorApp } = useCapacitor();

  return {
    isCapacitorApp,
    setStringValue,
    setBooleanValue,
    setNumberValue,
    setObjectValue,
    getStringValue,
    getBooleanValue,
    getNumberValue,
    getObjectValue,
    removeValue,
    clearAll,
    keyExists,
    PreferenceKeys
  };
}
```