Mobile Application Development

**SD6501**

Abhedika Singh

22300674

Table of Contents

[1. Component descriptions 2](#_Toc209821604)

[ExpensesProvider (src/storage/db.js) 2](#_Toc209821605)

[HomeScreen 2](#_Toc209821606)

[EditScreen 3](#_Toc209821607)

[SettingsScreen 3](#_Toc209821608)

[ThemeContext 3](#_Toc209821609)

[ChartCard 3](#_Toc209821610)

[DismissKeyboardView 4](#_Toc209821611)

[App / Navigation (App.js) 4](#_Toc209821612)

[2.The theoretical structure 4](#_Toc209821613)

[Features and principles of programming 5](#_Toc209821614)

[4. Analysis of development (constraints, strategies, challenges) 5](#_Toc209821615)

[5) Testing documentation 6](#_Toc209821616)

[6. App screenshots 8](#_Toc209821617)

[7. General format and presentation 14](#_Toc209821618)

[8. Conclusion 14](#_Toc209821619)

[9.References 15](#_Toc209821620)

# Component descriptions

## ExpensesProvider (src/storage/db.js)

I introduced a specialized provider which owns the whole set of expenses and does persistence. When an app is started it reads data found in AsyncStorage and after the initial load, it updates the data back whenever the list is updated. It reveals clean operations: add(expense), update(id, patch), remove(id) and a calculated total.

Enhancements in A2: I implemented the null-safe defaults which would not crash the screens when the context has not loaded yet, standardized the field used as the date by the filters and ensured that write-through persistence occurs only after the initial loading.

## HomeScreen

The main entry point is called the Home screen. It displays a header, total-spent card, a search box, date-range filters (Today/ Week/ Month/ All), a simple category summary chart, and expenses list with a quick delete action. It reads the list and total of the provider and the search query and range are local state based. Filtering Memoized responsiveness.

Enhancements in A2: I have added the range filters, the chart component, a keyboard-dismiss wrapper (tap the background to hide the keyboard), and complete dark-mode styling.

## EditScreen

Both Add and Edit modes are done in this screen. It has titles (title), amount, category chips and optional note. It verifies necessary title and positive amount and it verifies on delete. On submit invokes the add or edit of a provider, in edit mode there is a Delete button which invokes remove.

A2: There should be better validation, better labelling of buttons, and regular mode-based titles on navigation.

## SettingsScreen

There is a small screen, which switches between light and dark mode using a ThemeContext. Navigation theme is also varied so that there is consistent UI in both of the modes.

Enhancements to A2: New screen; theme choice is now everlasting and extends across the app.

## ThemeContext

Consolidates theme state (mode, resolved, toggle) to have uniform styling across screens and the navigation container.

Enhancement of A2: End-to-end reliable dark-mode.

## ChartCard

An easy to understand visualization of category based spending. It currently displays category totals and a SVG donut shape which is currently displayed as a placeholder to be converted into a pie chart. It gets filtered data already provided by HomeScreen.

Improvements in A2: Entirely new component; this is to provide more information than a simple list.

## DismissKeyboardView

A utility wrapper that closes the keyboard on a background tap that solves the tutor feedback issue and makes the small screen more usable.

Enhancement of A2: Novel utility in Home and Edit screens.

## App / Navigation (App.js)

One native stack (Home, Edit, Settings, Login), surrounded in SafeAreaProvider, ThemeProvider, and ExpensesProvider. NavigationContainer changes the theme depending on the mode.

Enhancements in A2: Reduced to single stack; theme sensitive navigation.

2.The theoretical structure   
  
Problem and objective: Create a straightforward, offline-first expense tracker that allows for quick additions and edits, fast searching and date-range filtering, and an easy-to-read summary of totals and categories.  
  
Scope and presumptions: local persistence only, one user, no back-end or accounts. Expo is used to run the app on iOS and Android.  
  
Design:  
1. Data layer: AsyncStorage + ExpensesProvider for CRUD and persistence.  
2. UI layer: Home, Edit, and Settings screens made up of tiny parts (ChartCard, DismissKeyboardView).  
3. Theme: Light/Dark ThemeContext; Navigation The container respects the theme.  
4. Navigation: To maintain simple flows, there is only one native stack.  
5. Data model: An expense has the following: date/createdAt, note?, category, amount, title, and id. JSON is used to serialise and deserialise data for storage.  
6. Design guidelines: Make it responsive, easy to use, and offline-friendly. To prevent runtime issues, use defensive/null-safe patterns and memoization for filtering.

Features and principles of programming   
  
Features that have been implemented:  
  
1. AsyncStorage for local persistence (load on start, save on change).  
2. Complete CRUD: Add, Read, Update, Delete (using date and search filters).  
3. Date-range filters: All, tMobile Application Developmenthis month, the last seven days, and today.  
4. A category summary chart (SVG doughnut placeholder) and a totals card.  
5. ThemeContext allows for dark mode; the navigation theme also changes.  
6. Safe-area handling; keyboard dismiss on background tap.  
7. Validation: a positive numerical value and the necessary title.  
  
Use of programming principles:  
  
1. UseState, useEffect, useMemo, and useContext are React hooks.  
2. Context API for less prop digging and shared state (data + theme).  
3. AsyncStorage uses JSON serialisation and deserialisation for persistence.  
4. Memorisation combined with pure assistance makes filtering effective.  
5. Defensive, null-safe destructuring to avoid crashes.  
6. UI/unit testing using the React Native Testing Library and Jest.

4. Analysis of development (constraints, strategies, challenges)

Constraints:

* Expo SDK 54 and peer dependency alignment (React 19, Reanimated, etc.).
* Mobile ergonomics: safe spaces, keyboard overlap, accessible tap targets.
* By assignment scope, Local/Offline only (no server sync).

Strategies:

* Install used npx expo-doctor and npx expo install to fix peer versions until checks pass (17/17).
* ExpensesProvider so screens are largely presentational with centralized business logic and persistence.

Challenges & resolutions:

* Windows long-path extraction error: LongPathsEnabled=1 enabled under HKLM\SYSTEM\CurrentControlSet\Control\FileSystem.
* Accidental PowerShell here-string parsing error in a test file: rewritten the tests as regular JS modules and cleared the Jest cache.
* useExpenses not a method / ThemeContext undefined in tests: mocked storage hook and theme; null-safe destructuring added to HomeScreen.

Expo peer dependency mismatches: aligned versions following expo-doctor guidelines; reinstalled compatible packages.

# 5) Testing documentation

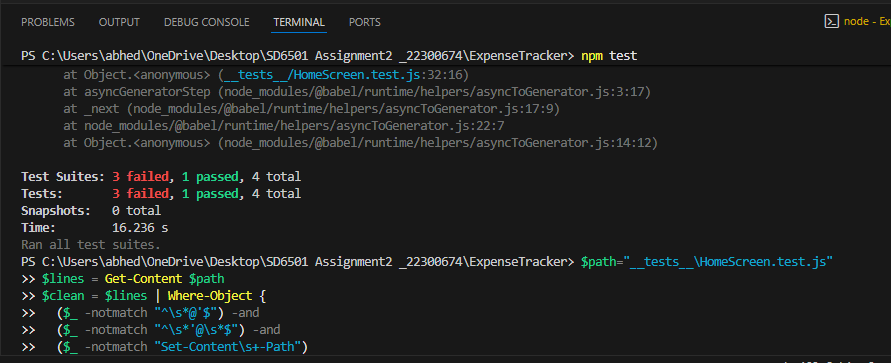
Tools: Jest with jest-expo as a preset; React Native Testing Library (RNTL).

Commands to run:

* npm test
* npm test -coverage (measures coverage summary)

Test cases & results (summary):

* T1 - Smoke: Smokes a trivial component; Renders (green).
* T2 - Home list: Mock useExpenses() and render Home; item titles are displayed and sum is equal (pass).
* T3 - Add expense: EnterEdit form; save; add called with numeric amount to be called which is to be parsed; back (pass).
* T4 - Update expense: Open Edit route.params save; update id + patch (pass).
* T5 - Delete expense: Click on Delete, verify; delete called with id (Pass).
* T6 -Settings theme: Make with mocked ThemeContext:screen renders, toggle renders without error and callable (pass).

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

# 6. App screenshots

A screenshot of a phone

AI-generated content may be incorrect.

Figure 1:Light mode

A screenshot of a phone

AI-generated content may be incorrect.

Figure 2:Dark mode

A screenshot of a phone

AI-generated content may be incorrect.

Figure 3:Range filter: at least Today and All views.

A group of blue and white buttons

AI-generated content may be incorrect.

Figure 4:ChartCard: category totals

A screenshot of a phone

AI-generated content may be incorrect.

Figure 5:Add Expense

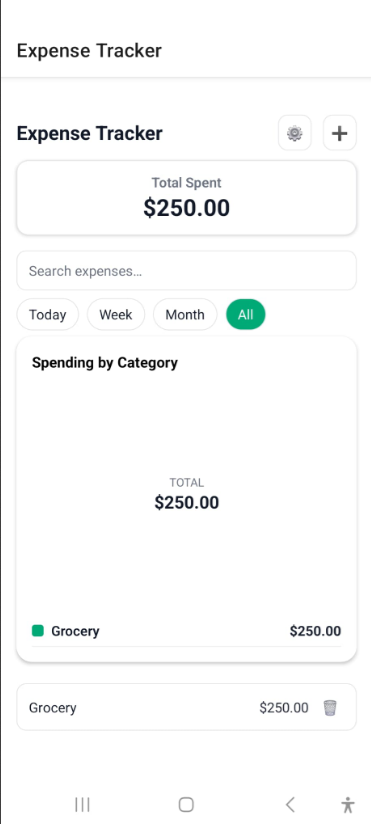


Figure 6:Delete confirmation dialog.

A screen shot of a phone

AI-generated content may be incorrect.

Figure 7:Settings screen: theme toggle visible.

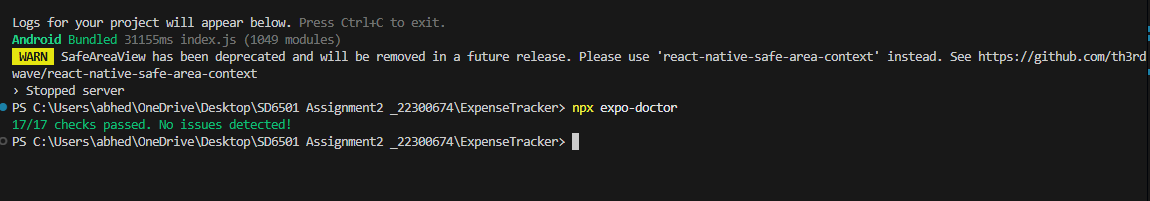


Figure 8:Expo doctor: npx expo-doctor showing “17/17 checks passed

7. General format and presentation

* Have regular headings and legible typeface.
* Close to every section that goes together have a screenshot close by with a caption.
* See original libraries utilized: Expo, React Navigation, AsyncStorage, react-native-svg, React Native Testing Library, Jest.
* Store snippets of code to the bare minimum and only relevant (e.g. commands to run tests).
* Export to PDF and save to be used in case of necessity.

# 8. Conclusion

The Expense Tracker satisfies Assignment 2: it stores data in the local storage with the help of AsyncStorage, offers a clean input experience and a list management, implements full CRUD, has useful filters and a categories overview, enables a dark mode, and has automated evidence-based tests. The codebase is not only modular but also reliable and I have appended the requested screenshots and test artefacts to facilitate the marking rubric.

# 9.References

* Expo. (2025). *Expo documentation* (Expo CLI, dependency validation, expo-doctor, SDK guides). Retrieved September 26, 2025, from <https://docs.expo.dev/>
* React Native Async Storage. (2025). *AsyncStorage documentation*. Retrieved September 26, 2025, from https://react-native-async-storage.github.io/async-storage/
* Meta Open Source. (2025). *React Native documentation* (Core components & APIs, SafeAreaView). Retrieved September 26, 2025, from https://reactnative.dev/docs/getting-started
* Th3rdwave. (2025). *react-native-safe-area-context* (Safe area handling). Retrieved September 26, 2025, from <https://github.com/th3rdwave/react-native-safe-area-context>
* React Navigation. (2025). *React Navigation docs* (Getting started, Native Stack, Theming). Retrieved September 26, 2025, from https://reactnavigation.org/docs/getting-started/
* Callstack. (2025). *React Native Testing Library documentation*. Retrieved September 26, 2025, from https://callstack.github.io/react-native-testing-library/
* Jest. (2025). *Jest documentation* (Getting started, React Native testing). Retrieved September 26, 2025, from [https://jestjs.io/docs/getting-started](https://jestjs.io/docs/getting-started?utm_source=chatgpt.com)
* Software Mansion. (2025). *react-native-svg* (SVG charts for React Native). Retrieved September 26, 2025, from <https://github.com/software-mansion/react-native-svg>
* Expo. (2025). *Icons in Expo* (@expo/vector-icons & expo-font). Retrieved September 26, 2025, from https://docs.expo.dev/guides/icons/
* Expo. (2025). *Managing dependencies with expo install* (SDK version alignment). Retrieved September 26, 2025, from https://docs.expo.dev/workflow/expo-cli/#installing-dependencies