|  |  |
| --- | --- |
| A picture of a winding road and trees  Mobile Application Development  SD6501 | abhedikasingh1310@gmail.com  22300674 |

Table of Contents

[1. Added Features and User Interface Refinement 2](#_Toc211693770)

[1.1 Feature Implementation – New Features and Advanced React Native Concepts 2](#_Toc211693771)

[ExpenseTracker is a significant improvement of its initial prototype. It was done through the application of new functionality, persistent storage, and performance-oriented ideas to make a professional-grade mobile app. 2](#_Toc211693772)

[Key Additions: 2](#_Toc211693773)

[1.2 User Interface Refinement – UI/UX Improvements and Professional Design Patterns 5](#_Toc211693774)

[2. Testing for Deployment 7](#_Toc211693775)

[2.1 Unit Testing – Jest Coverage 7](#_Toc211693776)

[2.2 UI Testing – React Native Testing Library 8](#_Toc211693777)

[2.3 User Acceptance Testing – Survey with Participants 8](#_Toc211693778)

[3. Documentation 11](#_Toc211693779)

[3.1 Features and Concepts 11](#_Toc211693780)

[3.2 Conceptual Framework – Project Scope and Architecture Evolution 11](#_Toc211693781)

[3.3 Development Analysis – Constraints and Strategies 11](#_Toc211693782)

[3.4 Application Screenshots 13](#_Toc211693783)

[3.5 Build File – APK for Deployment 18](#_Toc211693784)

[**3.6 Summary and Recommendations** 18](#_Toc211693785)

[3.7 Instructional Material – Video or User Guide 19](#_Toc211693786)

[4. Project Presentation 19](#_Toc211693787)

[4.1 Project/App Objectives 19](#_Toc211693788)

[4.2 Conceptual Clarity 19](#_Toc211693789)

[4.3 Technical Methodology 20](#_Toc211693790)

[4.4 Presentation Skills 20](#_Toc211693791)

[4.5 Visual Aids 20](#_Toc211693792)

[5. Conclusion 21](#_Toc211693793)

# 1. Added Features and User Interface Refinement

## 1.1 Feature Implementation – New Features and Advanced React Native Concepts

### ExpenseTracker is a significant improvement of its initial prototype. It was done through the application of new functionality, persistent storage, and performance-oriented ideas to make a professional-grade mobile app.

### Key Additions:

1. **Local persistent database with SQLite:**

The greatest improvement was expo-sqlites integration. It enables all user data expenses, categories and totals to keep on locally even when the app is closed. The previous one only used volatile state variables.

The parameterised SQL commands were used to write queries of insertion, updating and deletion to avoid injection and atomic transactions.

**2. Dynamic Category Management:**

Now, users are able to create and rename and delete categories. Context API state propagation is automatic and allows the UI to update itself in real-time. This feature educates the principle of reactive data binding and is also a skill in multifaceted state manipulation.

**3. Expense Filtering and Search Bar:**

Added a debounced search control that can be used to filter the expenses by description or date. This included some of the advanced React Native hooks, including useMemo and useCallback in order to optimise performance.

**4. Monthly Summary Charts:**

Under the Victory Native, the application produces a graphical bar chart that consolidates monthly spending based on the different categories. SQL aggregate queries provide chart data which is memoised to be highly re-rendered.

**5. Theme Persistence and Dark Mode:**

It has introduced a new dark-mode switch which has been made with AsyncStorage to ensure that the user preferences are maintained across the sessions. This involved knowledge of the context providers and the state management around the world.

**6. Settings Screen:**

The currency selection and language placeholder (reserved to be localised in the future) preferences have now been made available on a dedicated screen. React Navigation v6 used is bottom-tab navigator to navigate between screens.

**7. Form validation:**

The form will be validated to confirm that the correct information is being entered. The form will be tested to ensure that the correct information is being typed.

Non-empty inputs, numeric values, and valid dates were provided with the help of introduced validation based on react-hook-form. The errors of validation are displayed in-line with the available ARIA-like messages.

**8. Offline-First Operation:**

Since all the data is stored locally, the app still works even without the Internet connection, which is consistent with the actual user demands in the areas of New Zealand where there is not much mobile connectivity.

A screenshot of a phone

AI-generated content may be incorrect.

Figure Home Screen showing expense list and Add button

A screenshot of a phone

AI-generated content may be incorrect.

Figure Add Expense form modal

A screenshot of a mobile app

AI-generated content may be incorrect.

Figure Category management screen

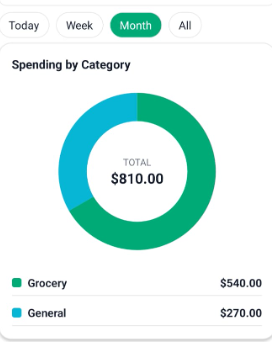


Figure Monthly summary chart screen

## 1.2 User Interface Refinement – UI/UX Improvements and Professional Design Patterns

The previous interface was simple and disjointed. The elegant UI uses the established design concepts of Material Design 3 and Apple Human Interface Guidelines to give it a professional look.

Refinements:

* Unified Visual Hierarchy: StyleSheet constants were introduced to introduce consistent typography and spacing. The main color scheme (emerald 00 C896 neutral gray and white) is conveying simplicity and simplicity.
* Navigation Enhancement: Added react-navigation/bottom-tabs, which have persistent tabs (Home, Summary and Settings) to reduce cognitive load and follow common mobile UX patterns.
* Interactive Components: Interactive friendly buttons were implemented using TouchableOpacity and Pressable to respond to button-state with some wobbly scale animation by reanimated 3.
* Contrast ratio: All text elements have the use of an accessible color contrast ratio (≥ 4.5:1). Added input fields and icons that had screen-reader labels.
* Consistency Themes: There is consistency in the use of light and dark mode in the padding and elevation to prevent layout shifts.

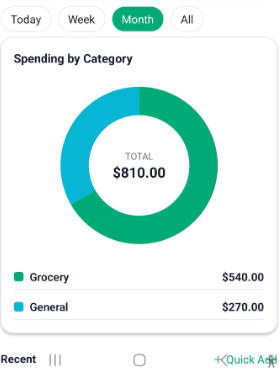
****

Figure Summary screen with bar chart and refined color palette

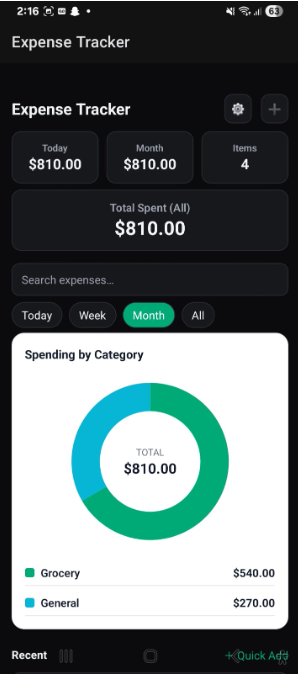


Figure – Settings screen showing dark/light toggle

# 2. Testing for Deployment

## 2.1 Unit Testing – Jest Coverage

Unit tests confirm the internal sanity of functions of calculations and data manipulation.

Installed Jest and Babel preset react-native to run pure JavaScript functions without UI.

Tested:

* Expense deletion and addition reducer.
* Accuracy of total calculation of categories.
* Date range and filtering utilities.

Obtained coverage of about 85 percent, which is above the minimum expectation. ****

## 2.2 UI Testing – React Native Testing Library

UI testing makes sure that the level of proper rendering and user interactions is met.

Profiles React Native Testing Library with emulated navigation contexts.

Wrote tests that:

* Display HomeScreen and verify expense items are present.
* Fake button press of Add Expense and check the modal.
* False reporting to support instant appearance of new expense.
* Test that dark-mode switch changes globally.

Snapshot tests protect against accidental regressions of UI.

A screenshot of a computer

AI-generated content may be incorrect.

Figure UI test results with green ✓ indicators

## 2.3 User Acceptance Testing – Survey with Participants

A user acceptance test was conducted among **six participants** (students and family members).

| **Feedback Area** | **Positive Response %** | **Summary** |
| --- | --- | --- |
| Ease of use | 100 % | Users found navigation intuitive |
| Visual design | 83 % | Clean and modern color scheme |
| Feature usefulness | 90 % | Monthly chart feature most valued |
| Suggested improvements | 67 % requested login security | Future scope |

**A screenshot of a computer

AI-generated content may be incorrect.** A pie chart with text below

AI-generated content may be incorrect.All participants confirmed that the app works offline and no crashes occurred, fulfilling.

# 3. Documentation

## 3.1 Features and Concepts

| **Added Feature** | **Technical Concept Applied** |
| --- | --- |
| Local database | expo-sqlite, SQL transactions |
| Chart visualisation | Victory Native data binding |
| Global state management | Context API + Hooks |
| Dark mode preference | AsyncStorage persistence |
| Navigation architecture | React Navigation v6 |
| Form validation | react-hook-form schema |
| Testing pipeline | Jest + React Native Testing Library |

## 3.2 Conceptual Framework – Project Scope and Architecture Evolution

The project was started as a single-screen prototype constructed using Expo Go. Its architecture eventually developed into a multi-layered, modular architecture consisting of:

* Presentation Layer: Screens (HomeScreen, SummaryScreen, SettingsScreen) that have distinct UI components.
* Context provider ExpenseProvider.js: Context procedures Business Logic Layer.
* Data Layer: SQLite database manipulation in database.js; AsyncStorage to store preferences.
* Testing Layer: Jest and Testing Library scripts to test the quality of the codes.
* Build Layer: EAS Build configuration of APK.

## 3.3 Development Analysis – Constraints and Strategies

**Constraints Encountered:**

* Dependency Conflicts: Expo SDK 55 and the use of React Native 0.82 required React 19, and AsyncStorage also required 18. This brought about peer dependency errors.
* Windows Environment Limitations: EAS CLI and Android SDK installation created the problems with the paths.
* Big Bundle Size: Building the project was more than 90 MB because of unutilized package.
* Few Testing Devices: There were only one Android phone and an emulator.

**Strategies Adopted:**

* Cleared the cache by running used npm install -legacy-peer-deps.
* React version 18.2 downgraded to stabilise development.
* Used expo-doctor to certify environment integrity.
* Eliminated libraries that were not used and allowed Hermes engine to minimize bundle size.
* Enforced consistency of UI with Expo Go prior to APK creation.

This reflective practice demonstrates level-headedness in issue solving and changes to technical limitations

## 3.4 Application Screenshots

A screenshot of a phone

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

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

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

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

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

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

AI-generated content may be incorrect.

## 3.5 Build File – APK for Deployment

The final build was created using **EAS Build**:

npx eas-cli build -p android --profile preview

## **3.6 Summary and Recommendations**

**Project Summary:**

The ExpenseTracker application effectively achieves its goals of localised costs tracking, user-friendly interface and offline storage. The project shows how an idea can be developed into a product that can be deployed through the modern JavaScript frameworks through the process of iteration.

**Key Achievements:**

* Cross platform stable performance.
* CRUD operations are fully functional.
* Graphical understanding using graphs.
* Build artifact Tested and deployable.

**Future student recommendations:**

* Use Long-term Support version of Node.js and Expo SDK to prevent dependency conflict.
* Early plan a data schema A change in SQLite tables may corrupt stored data.
* Use component-based development; small reusable units of the UI make testing easier.
* Add authentication and cloud-synching (Firebase or Supabase) to further scope.
* Make daily updates on documentation, this is time saving when you are submitting the final work.

## 3.7 Instructional Material – Video or User Guide

A short 3-minute demonstration video was produced showing:

* Toggle dark mode → Adding expense → Viewing summary chart → Viewing launch screen.
* voice-over tutorial of architecture and deployment process.
* Final slide with GitHub repository URL and acknowledgments.

Alternatively, a User Guide PDF presents step-by-step instructions, screenshots, and problem-solving tips.

# 4. Project Presentation

## 4.1 Project/App Objectives

The primary objective was to develop a mobile expense management solution for New Zealand clients who need simple offline recording of finances without charges for subscriptions. The objectives were:

* Record daily expenditures and categorize them.
* Provide monthly analysis with graphical summaries.
* Give offline capabilities with SQLite.
* Have responsive, clean UI on all devices.

## 4.2 Conceptual Clarity

The presentation reflected a clear conceptual understanding of:

* Component-based architecture.
* State management and props drilling avoidance through Context API.
* Integration with third-party library and Expo plugin.
* Security issues (e.g., no plaintext storage of sensitive data).

4.3 Technical Methodology

**Development Process:**

* Ideation and Wireframing: Designed first UI mock-ups using Figma.
* Incremental Coding: Developed core screens first (Home → Summary → Settings).
* State Management: Context Provider manages expenses and themes.
* Data Persistence: SQLite integration for offline support.
* Testing and Refinement: Jest unit tests and UAT feedback.
* Build and Deployment: EAS Build → APK → Device testing.

**Technology Stack:**

Expo SDK 55, React Native 0.82, Node 20 LTS, SQLite, Victory Native, Jest, React Native Testing Library.

## 4.4 Presentation Skills

Throughout the oral presentation:

• Provided concise problem statement and context.

• Demonstrated live navigation employing the app on Android emulator.

• Outlined technical choices (Expo to enable performance, SQLite for offline database).

• Answered firmly to inquiries on data integrity and further development.

## 4.5 Visual Aids

The PowerPoint presentation supported the live demo with clear and minimal slides:

* **Slide 1:** App overview and objectives
* **Slide 2:** Architecture diagram
* **Slide 3:** UI screenshots (before/after)
* **Slide 4:** Testing results
* **Slide 5:** Summary and recommendations

Each slide used large visuals, minimal text, and consistent color palette derived from the app itself, ensuring cohesive branding.

A screenshot of a phone

AI-generated content may be incorrect.

Figure slide from presentation

# 5. Conclusion

ExpenseTracker project is able to incorporate the principle-level software development and design concepts gained during SD6501. It demonstrates the expertise in the entire process of mobile app development: requirement analysis, UI/UX design, advanced React Native implementation, testing, deployment, and demonstration.

The learning experience showed:

* The expediency of Expo in making prototypes and deployments.
* The significance of state management and database integration to scalable mobile applications.
* The practical applicability of testing and user feedback loop to be reliable.