Ministry of Science and Higher Education of the Russian Federation
"Ural Federal University named after the first President of Russia B.N. Yeltsin"
Institute of Radio electronics and Information Technologies – RTF

Restaurant Choser

Report Project

Student : AYAD Seif Eddine

Group: RIM-140930

TABLE OF CONTENTS

Introduction

This report presents a comprehensive analysis of the mobile application project provided. The primary objective of this phase was to systematically identify all existing errors — whether syntactic, runtime, or logical — and apply appropriate corrective actions. Additionally, enhancements were introduced to improve the application's functionality, usability, and maintainability.

The process involved a methodical review of the codebase, rigorous testing of application features, consultation of official package documentation (particularly for **expo-sqlite** and **@react-navigation**), and critical evaluation of development practices. This report is structured to first explain the error identification process, then to document each encountered issue alongside the solution applied, and finally to summarize the enhancements introduced beyond basic error correction.

1. Error Identification Process

The following structured methodology was employed to ensure thorough identification and understanding of all existing issues:

1.1 Environment Setup and Initialization

We cloned the repository locally using Git, installed all project dependencies with npm install, and started the development server using npm start to launch the application.

1.2 Manual Testing Procedures

We cloned the repository locally using Git, installed all project dependencies with npm install, and launched the development server using npm start. After successfully starting the application, we navigated through the user interfaces, including buttons, tab navigations, and different screens. Throughout the process, we closely monitored the console for any red-box runtime exceptions or yellow-box warnings, which typically signal underlying code issues in React Native applications. Additionally, we engaged in various typical user activities, such as screen switching, button tapping, and general navigation, to simulate real-world usage scenarios and uncover potential logical or functional inconsistencies.

1.3 Codebase Static Analysis

We conducted a manual review of the source code, paying particular attention to several key areas: the correct usage of React Native components to ensure structural integrity; proper validation of props using PropTypes along with the correct application of defaultProps; the configuration of the navigation structure and the accurate passing of parameters between tab screens; and finally, the implementation of SQLite database operations, verifying their alignment with best practices recommended for the **`expo-sqlite`** package.

1.4 Cross-Referencing with Official Documentation

We verified the integration patterns against the official documentation for **`expo-sqlite`** to ensure correct database transaction handling and query execution, and for **`react-navigation`** to confirm that navigators and screen transitions followed the updated API guidelines.

1.5 Systematic Documentation

All identified anomalies were recorded immediately, including error messages, conditions under which they occurred, and their apparent impact on user experience or code stability.

2. Detailed Error Identification and Solutions

Below is a categorized and detailed breakdown of each issue we detected during the process, alongside the steps taken to resolve them.

2.1 Missing or Improper Application of Prop Values

Component Affected: `CustomButton`

Error Description: The `CustomButton` component included a `width` prop with PropTypes and a default value set to `100`, but this value was never actually applied to the button's visual style.

Impact:

- Buttons rendered with inconsistent or default sizing behavior.
- Deviation from intended UI design specifications.

Root Cause: While `width` was defined and passed as a prop, it was omitted from the `TouchableOpacity` `style` attribute.

Solution:
Amend the `TouchableOpacity` element to dynamically include the `width` prop within the applied styles.

```
<TouchableOpacity
  style={[buttonStyle, { width: width }]}
  onPress={onPress}
  disabled={disabled}
>
  <Text style={textStyle}>{text}</Text>
</TouchableOpacity>
```

Outcome After Fix:

- Buttons now properly respect the specified width prop value.
- Visual consistency improved across different parts of the application.

2.2 Tab Navigation Positioning Inconsistency

Component Affected: Tabs Navigator (material top tab navigation)

Error Description: The application attempted to control the position of tabs (top or bottom) based on the platform. However, tabBarPosition is not supported for createMaterialTopTabNavigator on iOS — it is fixed to "top."

Impact:

- Tab position on iOS could not be changed, leading to a mismatch between code intent and runtime behavior.
- No visible crash, but misalignment with the intended UX.

Root Cause: **createMaterialTopTabNavigator** always renders tabs at the top; the conditional logic intended to place it at the bottom on iOS is ineffective.

Solution:

- **Alternative 1 (Recommended):** If bottom tabs are truly desired on iOS, use createBottomTabNavigator from @react-navigation/bottom-tabs instead.
- **Alternative 2:** Accept that Material Top Tabs always appear at the top and remove unnecessary tabBarPosition condition.

2.3 Static Asset Loading Errors

Component Affected: Icons in Tab Screens (`require` statements)

Error Description: The `require` path for icon assets such as `'../assets/icon-people.png'` may fail if the relative path is incorrect or the asset is missing.

Impact:

- Placeholder images or missing icons rendered in the tab bar.
- Visual degradation of UI.

Root Cause: Static assets must exist at the specified paths relative to the compiled JS bundle, and naming must match exactly (case-sensitive).

Solution:

- Confirm that each file (e.g., `icon-people.png`, `icon-decision.png`, `icon-restaurants.png`) exists in the `assets/` directory.
- Correct file paths if necessary.

Best Practice:

- We use `Asset.fromModule()` (from Expo) for more reliable asset handling in larger applications.

Outcome After Fix:

- All icons displayed correctly within tab navigator.

3. Enhancements Introduced

In addition to resolving the identified issues, several enhancements were made to improve overall code quality and user experience:

3.1 UI/UX Improvements

We dynamically applied button widths to improve responsiveness, enhanced button styling by using centralized and scalable styles, and cleaned up the tab navigator to remove misleading platform-specific logic.

3.2 Code Quality Enhancements

We enforced strict PropTypes validation across all components, removed redundant properties that had no effect on runtime, and structured imports consistently to improve overall maintainability.

3.3 Platform Consistency

- Ensured visual behavior is identical across Android and iOS (tabs always at the top).
- Used Platform.OS effectively only where truly impactful (e.g., padding adjustments).

Conclusion

Through a comprehensive process of error identification and enhancement, we significantly enhanced the application's stability, maintainability, and overall user experience. By applying industry best practices and refining platform-specific behaviors, the project now features a cleaner, more organized structure that is well-suited for future enhancements, scalability, and deployment.