***Product Documentation***

**1 Introduction**

The “Where Is” application is designed and created to help users create items and categories and locate rarely used or misplaced items. By storing names, descriptions of where items are placed, photos and GPS coordinates, it will be easy for a forgetful person like me to find my items. This document will include the functional and non-functional requirements of the application, user interface design, navigation strategy, data sources, external dependencies and testing strategy, ensuring compliance with all specified requirements. The document will help you understand the functionality as well as the libraries I used.

**2 Functional and non-functional requirements for the application**

The following are all the functionalities fully implemented in WhereIs application, all the required functionalities have been implemented.

2.1 Functional Requirements

R1.1 Entry Screen

* R1.1 The entry screen includes 2 buttons is “Add new” and “List items”, I implemented using Expo Router stack navigation to the Record Creation screen, it show in the file name AddItem.js and the href is /addItem.
* R1.2 The “List items” button will navigates to a FlatList of WhereIs – items, it show in the file name ListItem.js and the href of this file is /listItem.

R2.2 Record Creation

* R2.1 The Record Creation screen (AddItem.js) provides a form for entering the item name, description, also taking a photo and capturing GPS coordinates using expo-location and expo-image-picker.
* R2.2 It will ensure input validation of name and description are mandatory, displaying error messages prompt if one of the fields is missing or both.
* R2.3 I used expo-secure-store to save the record with all the details provided.
* R2.4 Error messages are shown if storage fails, using Alert.alert.

R3.3 Flat list screen

* R3.1All the saved record will be displayed in the FlatList page (ListItem.js file).
* R3.2 Each item shows the name and description, and a thumbnail of the photo optimized using expo-image.
* R3.3 Stack navigation links list items to the View One Item screen (ItemDetails.js) href is /items/[id].
* R3.4 A back button with arrow icon navigates to the entry screen.

R3.6 View one item-screen

* R3.6.1 The ItemDetails.js screen displays the item’s name, description, GPS coordinates, and photo and also have a view map button (when you click it will deliver you to google map app and have a picture for you location depend on GPS) and edit, delete button too.
* R3.6.2 A back button navigates to the FlatList.

R4.4 Storage of Records

* R4.1 Records are stored locally using expo-secure-store with a way to encrypt and securely store key-value pairs locally on the device, ensuring data security without a remote backend.
* R4.2 Each record includes name, description, photo URI, and GPS coordinates and id.
* R5.3: I ensure data integrity during the save process, all inputs are validated before saving. The app uses try/catch to catch any errors during storage operations. If an error occurs, the app notifies the user with a meaningful message and does not save incomplete or corrupted data.

R6.6 Search Functionality

* R6.1 A search bar in ListItem.js so you I search your items
* R6.2 Supports exact match for the name and partial match.
* R6.3 Search results will display the card of the item which have name, description and thumbnail, navigable to full details.
* R6.4 The result will display immediately every time you type 1 letter, it will list all items related to the name you search.

R7.7 Record Updates and Deletion

* R5.5 In the detail of each item will have edit button, when you click it will lead you to EditItem.js screen allows updating name, description, and photos and gps.
* R7.1 Deletion is supported in ItemDetails.js with a confirmation prompt.

R8.8 Photo Management

* R8.1 Photos are captured as JPEG using expo-image-picker.
* R8.2 Photos are optimized to 800x600 pixels at 60% quality using ImagePicker quality 0.3 in Expo Go. This will make sure the picture is not too heavy, and it will make the pictures load faster.
* R8.3 Photos are previewed, and you can adjust before saving using expo-image, even if you already have the picture in preview but still want to change, you just need to click take photo button again and take a picture

R9.9 Usability Features

* R9.1 Clear error messages are shown for invalid inputs with Alert.alert.
* R9.2 The UI is intuitive, with consistent navigation using Expo Router.
* R9.3 Even Deletion or Edition requires confirmation prompts.

R10.10 Security

* R10.1 Data is encrypted, decrypted and stored in expo-secure-store.
* R10.2 Updates and deletions require user confirmation.
  1. **Non-Functional Requirements**
* **Performance**: Image loading is optimized using expo image caching and FlatList tweaks. In addition, all images used are quality optimized to avoid slowing down the application and slowing image rendering.
* **Security**: Encryption ensures data protection (R10.1).
* Usability: Intuitive UI and clear error messages enhance user experience (R9.1, R9.2).
* **Compatibility**: Fully compatible with Expo Go, using Expo SDK modules. Can use in phone with version iOS 14.0 or later and Android 6.0 (API Level 23) or later.

**3 User interface design and navigation strategy explained**

**3.1 User interface desgin**

The WhereIs app has a clean, intuitive user interface with a blue color theme (#63D8FE background) and a mix of black and white text and box borders. All are combined in a simple way between black and white and blue to make the user feel comfortable when looking at it. The main components include:

* The main screen has two buttons (“Add New”, “List Item”) styled with TouchableOpacity and MaterialIcons and at the top is the app logo.
* Create Record (AddItem.js): A form with TextInput for name and description, photo preview with expo-image and GPS display. The buttons use #1CA0DC for actions. There will be three buttons: take photo, get gps and save item.
* Flat List (ListItem.js): A flat list with a tag (#eeeeee background) showing the name, description and 100x100 thumbnail. The search bar uses TextInput with a magnifying glass icon. And the tags are also considered buttons because I used (TouchableOpacity) so that when you click on it, it can take you to the item details page
* Item Details (ItemDetails.js): Shows the name, description, GPS and 390x330 image. Includes buttons to view the map, edit and delete. Along with that are the prompts that will appear when you confirm whether you want to delete or edit the item. The prompts will clearly show the information and will be related to the action you are doing.
* The styles use StyleSheet.create with Tagess-Reg and Space-Mono fonts,Playwrite-RO to ensure readability and consistency. All are designed to help readers easily read and visualize the app. All the alear will help users understand what you should do when you do something wrong

**3.2 Navigation Strategy**

The application uses expo-router to navigate the stack:

* Input screen: The root route (/) navigates to /addItem or /listItem.
* Create record: after clicking /addItem, you will go to the record creation page, then after filling in all, you click save item, which will take you back to the main page, there is also a back button displayed on the app as an arow icon.
* Flat list: after clicking /listItem, you will go to the flat list page, here there will be cards displaying item information, when clicking on the card, navigate to /items/[id] to know the details of each item and both pages have a back button displayed on the app as an arow icon, if in the flat list, you will be taken back to the main page, and if in the item detail page, you will be taken back to the flat list page.
* Item details: when you go to /items/[id] (called the item details) you will have a navigation button back to /listItem or /items/editItem to update. One thing worth noting is that when you click on the edit button but then you don't want to change it anymore, you just click the back button like any other page and your item will be kept intact and will come back to itemDetails of that exact item
* Seamless navigation, with useRouter and useLocalSearchParams passing the item data as JSON.

**4 Data Sources**

The WhereIs app is made up of the following data sources:

* Local storage (expo-secure-store): Items are stored locally in a JSON array encoded under the key 'items'. Each item consists of an id, name, description, photoUri and
* gpsCoordinates.
* Device camera (expo-image-picker): Images taken with the device camera, compressed to quality 0.3 (addItem.jsx) or 0.5 (editItem.jsx) and stored as URIs.
* Device GPS (expo-location): GPS coordinates (latitude, longitude) retrieved through the device's location services, stored with the items.
* Static assets: Including fonts (PlaywriteRO-Regular.ttf, SpaceMono-Regular.ttf, TagesschriftRegular.ttf), logos (logo.png), and default images (default.png and nopicture.png) for non-image items.
* No external APIs or databases are used, ensuring offline functionality and data privacy.

**5 External Dependencies**

The app uses the following libraries to code and testing:

* expo: Core SDK for React Native development, version 51.0.8.
* react: React framework, version 19.0.0.
* react-native: React Native framework.
* expo-router: File-based navigation.
* expo-font: Load custom fonts (Playwrite-RO, Space-Mono, Tagesschrift-Regular), version 12.0.10.
* @expo/vector-icons: Provide icons (MaterialIcons, AntDesign, MaterialCommunityIcons, FontAwesome6), version 14.0.3.expo-image-picker: Cameraaccessforphotocapture, version 15.0
* expo-location: Retrieve GPS coordinates.
* expo-secure-store: Encrypted local storage.
* react-native-maps: Displays a map for the item location.
* react-native-safe-area-context: Ensures UI compatibility with notched devices.

Testing libraries:

* jest: Testing framework
* jest−expo: A pre-configured preset for Jest provided by Expo that makes it easier to write and run tests in the Expo environment (npx jest)
* @testing-library/react-native: Used to test the interface (component) in the application. Used to check if a component is displayed correctly, works as expected
* @testing-library/jest-native: Used to write expect(...) lines that are easier to read and understand. Helps make tests easier to write, read, and maintain.
* All dependencies are compatible with Expo SDK 51 and Expo Go. The rest are libraries that are automatically installed when I download the project vs Expo. I also use Expo go and emulator to test the app, Expo go is IOS on my iphone and emulator is Android on my windows laptop.

**6 Testing strategy used**

* Testing strategy, the files to be tested are AddItem.test.js, listItem.test.js, itemDetails.test.js and editItem.test.js, ensuring strong validation of functionality and usability:
* Framework: Jest with jest-expo preset, integrated with @testing-library/react-native for component rendering and interaction.
* Dependencies (expo-secure-store, expo-image-picker, expo-location, exporouter, react-native-maps, Links) are mocked to isolate tests and avoid native module issues.

Test coverage: Target 80–90% coverage tests:

* UI rendering: Verify text, inputs, buttons and icons display correctly (e.g. ”Add new item”, ”List all items”).
* Input Validation: Ensure required fields (name, description) trigger warnings if left blank.
* Storage: Test safe storage and retrieval of items, including error handling.
* Navigation: Validate navigation for add, list, and detail screens, edit screens.
* Capture: Test optimized quality capture and permission handling.
* GPS: Verify coordinate retrieval and display.
* Search: Test filtered by name and description.
* Edit/Delete: Ensure edits save changes and remove confirmation prompts.
* Error Handling: Tests simulate permission denials, storage errors, and invalid input, verify warning messages.
* Setup: jest/setup.js extends Jest with @testing-library/jest-native for native component assertions.
* Implementation: Tests run through ‘npx jest –coverage‘, ensuring reliability on iOS 14+ and Android 6+.
* This strategy prioritizes functional correctness, usability, and error resilience, consistent with the application requirements.

**7 Conclusion**

WhereIs app has fully met all functional and non-functional requirements (R1.1 to R10.2). Intuitive user interface, powerful navigation, secure storage and optimized image processing make the app load faster from image to function. In conclusion, the app is easy to look at, easy to use and has been thoroughly tested to make users feel comfortable when using it and not worry about forgetting where they put their things.

**8 Note**

There are some things I want to explain about my app:

First, I used eas build to create apk file and a QR code also to download the app to test. But all is for android only I can't do it for ios because I have trouble with apple developer account. But if you run it on your emulator or expo go I guarantee it can be used on both IOS or Android. And when i test it, mine development sever still need to run so i guess even you can download it to your phone but still need my project to run. But anyway, I will still leave my development link for you (but it just works when I run it in my vscode). And like I said I use expo go in my iphone and try install apk or QR code in android phone so you might see some screenshots in different operating systems.

Second, I will upload all screenshots **in PNG-format** of every view of app taken from your phone screen but won’t have a pop up aleart like successfully add item or confirm prompt to edit or delete (when you use it in emulator still have of course)

And finally, if there is something wrong with the app, I can always make a video and send you I can confirm the app works fine with all requirements

**9 Apk file and QR of App**

**A qr code on a white background

AI-generated content may be incorrect.**

[**https://expo.dev/artifacts/eas/ocRupNKEaRvUKzScUe28hg.apk**](https://expo.dev/artifacts/eas/ocRupNKEaRvUKzScUe28hg.apk?fbclid=IwZXh0bgNhZW0CMTAAYnJpZBExU01aY3BRWGxSeWprSEtsVgEeC77UPdQPMbzgcTUthIwogyESvQTAOF1lz1GLnZmYD8RBgpsaZ0DCMHv4j9k_aem_f_RenKZuupo7PIOQ3A6Xig)

Development build link: exp+whereisapp://expo-development-client/?url=https%3A%2F%2Feyfbj5c-devpro1-8081.exp.direct