***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 provided details.
* 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 Functional Requirements**

**3.2 Non-Functional Requirements**

**4 External dependencies of the app (libraries used)**

**5 Data sources used**

**6 Testing strategy used**

**7 Conclusion**