# Overview

## Firebase

### Config

This file is where all the background settings and information are stored and exported from firebase for the rest of the program to use.

### DatabaseFunctions

Here we have some basic functions that are called from “Screen” files to interact directly with the firestore database. This file holds two methods: addItemToDatabase, and RemoveItemFromDatabase. Both of which are exported for use in other files.   
  
addItemToDatabase takes in product information of zone, shelf, upc, and quantity and adds the information to the inventory database in their respective fields. This method is called by the DataEntryScreen.  
  
RemoveItemFromDatabase takes in the parameters of zone, shelf, upc, and quantityToRemove and uses it to change the quantity of a specific product upc with specified zone and shelf fields. It then provides confirmation that the quantity was changed, removed, or if the product was not found in the location. This method is called in the removalScreen

## Navigation

### AppNavigator

This file oversees routing the user through the various screens in the app. Its only method is AppNavigator where it sets the initial start screen to the LoginScreen and is then called by other screens to send the user to the desired screen when desired. This file treats the navigating through the screens as a stack.

## Screens

All the screens will contain a “styles” file that exports basic information on how buttons, text in buttons, view containers, and headers will be viewed on their corresponding screen. Such information includes colors, font sizes, margins, padding, text styles, and other style settings.

### AccountScreen

This file displays the Account screen where a user can see their personal account information of email, full name, and profile picture, if provided by the user. It also has a method of handleImagePicker that allows the user to set a profile picture. At the top of the screen is the name of the screen and a back arrow to return to the Home screen. There is also a Logout button to enable the user to log out and then navigates the user to the Login screen

### DataEntryScreen

This screen is the purpose of the entire App. The display is from top to bottom and is as follows: The top left features a back button that will send the user to the previous screen, which will use the AppNavigation file to direct to the Home screen. The title of the screen is displayed across the center top. Proceeding down the center a barcode button that will open a camera so that the user can scan upc barcodes with their camera phone. Followed by input fields of Zone, Shelf, UPC, and Quantity for the user to input the respective information. Last are two buttons titled “Exit Shelf” and “Exit Zone” that will exit the shelf or zone the user is currently in. If the barcode button is pressed, then the screen changes where below the Shelf input field is a button titled “Close Camera” that closes the camera. The camera can also be closed by pressing the barcode button again.  
  
Upon entering the Data Entry screen, the user input is automatically directed to Zone and a numeric input pad pops up with the Enter key being replaced with a Next key. After entering Zone information and pressing Next the user is automatically sent down to the next input field. The process is the same for the Shelf field. If the camera is open the app will wait for the user to scan a barcode. When a barcode is scanned the camera will automatically close and input the barcode into the UPC field and automatically proceed to the quantity field. If the camera is closed, then the user can enter a upc manually and press the Next key to proceed to the quantity field. Once the user gets to the quantity field, the “Enter” key is changed into a “Done” key. Once information is entered and “Done” is pressed the handleAddUpc method is called which uses the addItemToDatabase method imported from DatabaseFunctions to add the zone, shelf, upc, and quantity information of the product to the database. The values for upc and quantity are reset to being empty and the user is automatically sent to the upc field for more information with the Zone and Shelf field unchanged. When the user wants to enter a different Zone or Shelf Field, they can either press the input fields or press one of the two buttons to simultaneously clear all data from the fields below it and be directed to enter information from that field by the handleAddShelf method or handleAddZone method, respectively. Should a user accidentally reach the end of information inputting and forget to fill out one of the fields, the handleAddUpc method will alert the user that all fields need to be filled, and the method will not store incomplete information in the database. To exit the Data Entry screen, simply press the back arrow or press back on your phone.

### GenerateReportScreen

The Generate Report Screen is a key component of the application that allows users to generate reports based on a specified zone and shelf range. It provides a user-friendly interface for inputting range parameters and dynamically fetching and displaying relevant inventory data from the Firestore database. The screen leverages the Firebase Firestore for data retrieval and renders the information in an organized and readable format.

The header section features a navigation button that allows users to return to the Home Screen. This is done by using React’s TouchableOpacity onPress prop, which then calls React Navigation’s navigate function. It also includes a title, “Generate Report Screen,” that provides clear identification of the screen’s purpose.

Users can input the start and end values for both the zone and shelf ranges. These input fields are styled for clarity and consistency, with placeholders indicating the purpose of each input. The values are set and saved in the following variables: zoneRangeStart, setZoneRangeStart, zoneRangeEnd, setZoneRangeEnd, shelfRangeStart, setShelfRangeStart, shelfRangeEnd, setShelfRangeEnd.

The screen dynamically fetches the inventory data from the Firestore database based on the specified zone and shelf ranges. The retrieved data is organized in a structured format, displaying Zone, Shelf, UPC, and Quantity. The data is sorted first by shelf and then by zone for a well-organized presentation.

React’s FlatList component efficiency renders the retrieved data in a scrollable list. Each row of the list displays information for a specific inventory item, including its Zone, Shelf, UPC, and Quantity.

### HomeScreen

After logging in, the Home Screen is the main entry point of the application. It provides users with essential information and quick access to various features.

The Home Screen fetches user information, including the user’s full name and profile picture from Firebase upon component mount. If the user is authenticated, then their details are retrieved and displayed in the header section.

The main grid container features buttons organized in rows. Each button corresponds to a specific functionality, providing a seamless navigation experience for the user. Each button contains a picture and a label to help users navigate to the correct destination.

The Home Screen source code was structured to provide a visually appealing and user-friendly interface. It effectively integrates with Firebase to display user information dynamically and facilitates seamless navigation to different app features.

### LoginScreen

The Login Screen serves as the initial point of user authentication, providing a secure entry into the application.

The Login Screen features a visually appealing background with a cover image, creating an immersive user experience. This was done utilizing React Native’s ‘ImageBackground’ component.

The screen includes input fields for email and password, allowing users to enter their credentials. Upon providing valid details, users can log in, triggering the ‘handleLogin’ function, which is done through React’s ‘TextInput’ component.

Users can navigate to the registration screen by pressing the “Sign up” link. The ‘onSignUpPress’ function handles the navigation.

Additionally, the Login Screen is styled with a combination of colors, button styles, and text formatting to provide a visually appealing and cohesive user interface. These styles can be found in the Login Screen’s styles.js file.

### RegistrationScreen

The Registration Screen is a crucial component of the application, providing users with the ability to create a new account. It presents a user-friendly interface for inputting necessary details, including full name, email, password, and confirmation of the password. The screen incorporates Firebase authentication and Firestore database integration to securely manage user accounts.

The screen features a prominent title “Registration” at the top, providing users with a clear context of the purpose of the screen. This is exhibited by the page’s header.

Users are prompted to enter their full name in a designated input field. The input is styled for clarity and consistency. An input field is also provided for users to enter their email address. This ensures a unique identity for each user. Lastly, users can securely input their chosen password. The input is configured as a secure text entry to protect sensitive information. This was done using React’s TextInput’s secureTextEntry prop.

The “CREATE ACCOUNT” button is prominently displayed at the bottom of the screen. When pressed, it triggers the registration process, creating a new user account using Firebase authentication and adding user details to Firestore database. This functionality is handled by the onRegisterPress function.

The footer includes a helpful message indicating that users can log in if they already have an account. A link is provided to navigate to the Login Screen, creating a seamless user experience. The link is React’s Text component with the onPress prop that calls the onFooterLinkPress function.

### RemovalScreen

The Removal Screen in the mobile application removes items from the database. It provides a user-friendly interface for users to input details, including the UPC (Universal Product Code), Zone, Shelf, and Quantity to remove. Additionally, the screen incorporates a barcode scanner to streamline the UPC input process.

The header section of the Removal Screen displays the screen’s title (“Removal Screen”) and includes a back button, allowing users to navigate back to the Home Screen.

Users can manually input the UPC of the item they want to remove. The input fields support numeric values only and are presented in a visually appealing style. This was done using React’s TextInput component.

To simplify the process of entering the UPC, a barcode icon is provided. When pressed, it activates the device’s camera to scan the barcode, which utilizes expo’s BarCodeScanner component. The scanned UPC is then displayed in the corresponding input field.

Users are prompted to input the Zone and Shelf information for the item they wish to remove. These input fields ensure accurate identification of the item’s location. Which also utilizes React’s TextInput component.

A dedicated input field allows users to specify the quantity of items they intend to remove. The field supports numeric input only, as set in the source code.

The “REMOVE ITEM” button triggers the removal process. Before initiating the removal, the application validates that all required fields (UPC, Zone, Shelf, and Quantity) are filled out. If not, an alert prompts the users to complete the missing information.

When the barcode icon is pressed, a modal overlay containing the device’s camera and a “Close Camera” button appears. This feature enhances user experience by providing an alternative method for entering the UPC. Which is handled by the function handleBarcodeIconPress.

The screen’s style and features a clean and intuitive design. A consistent color scheme, clear input fields, and well-designed buttons contribute to a visually appealing and user-friendly interface. The use of icons enhances the screen’s aesthetics and provides clarity to users about the functionality associated with each element. Styles can be found in the Removal Screen’s styles.js file.