# SP3-GET OUT!

Software Design Specification

OCTOBER 12, 2022

4850-01 SENIOR PROJECT

Fall 2022

Sharon Perry

# Contents

# 1 Document Management

## 1.1 Contributors

| Role | Name |
|---|---|
| Project Owner | **Sharon Perry** |
| Team Lead | **John Hussey** |
| Frontend Development | **Ryan Kim** |
| AI Development | **Nick Nguyen** |
| Backend Development | **Colin Allen** |
| Documentation | **David Shipman** |

## 1.2 Version Control

| Date | Version | Author | Section | Amendment |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# 2 OVERVIEW

## 2.1 General Overview

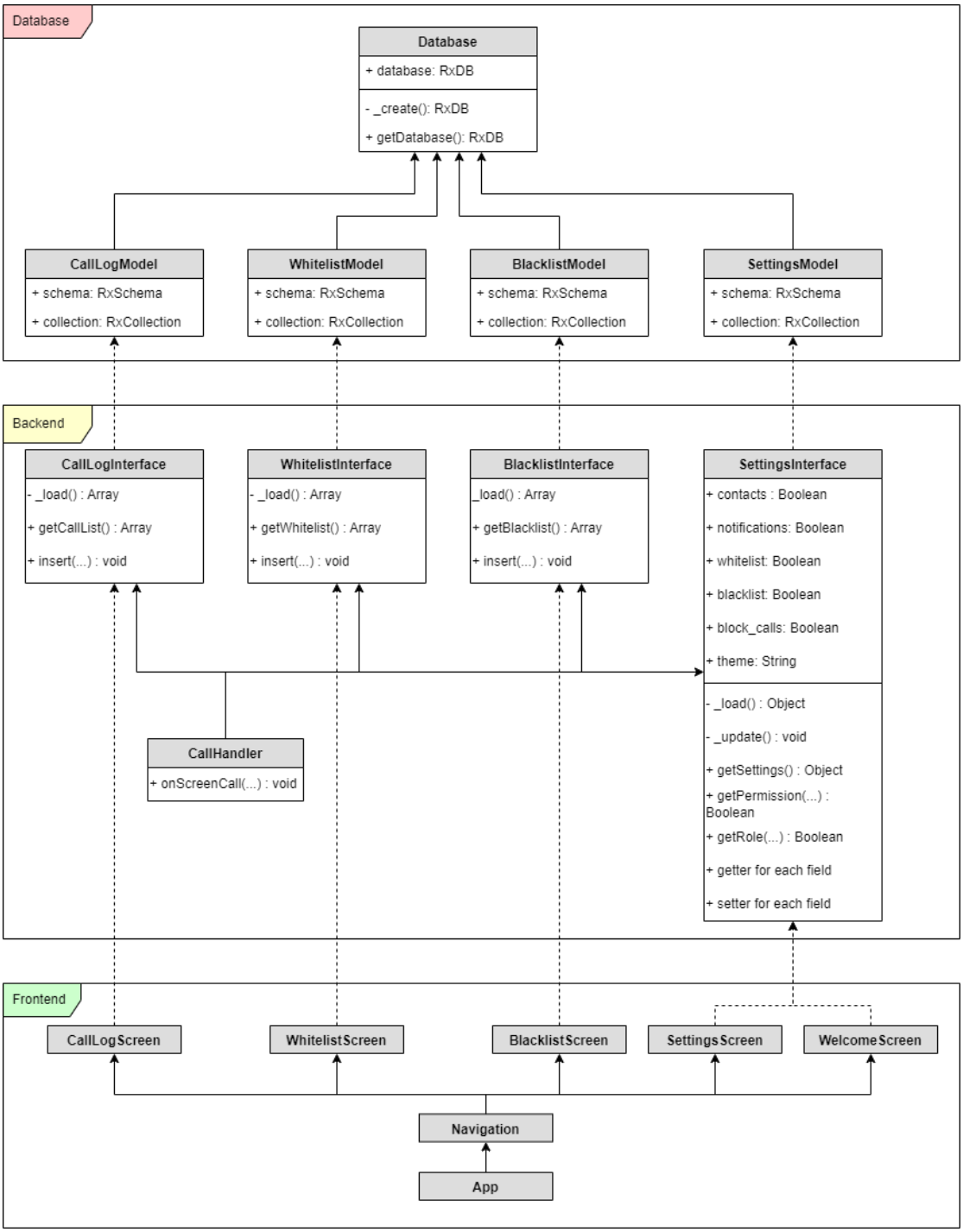
*The GetOut app is designed to intercept and block unwanted calls based on a combination of user configuration and heuristic automation. This includes detecting incoming calls and checking the phone number against a whitelist and blacklist. The whitelist is a list of whole or partial phone numbers to allow through, and it contains the user's contacts plus anything manually added by the user. The blacklist is a list of whole or partial phone numbers to block, and it contains anything manually added by the user. A priority system will determine whether to use the blacklist or whitelist for a given number.*

*Calls will be intercepted in the background, and a silent notification will be sent to the user if a call is blocked. All calls will be displayed in a call log, which will have options for the user to block or unblock callers manually. A settings menu will allow users to view and edit the whitelist and blacklist directly, enable or disable features, and change the app's theme.*

## 2.2 Design Strategy

*The system was designed using a bottom-up approach, and it consists of 3 layers: database, backend, and frontend. The database layer connects to the database and performs operations on it. The backend layer contains convenient methods to interact with the database, adding another layer of abstraction on top of the database layer. It also handles background tasks, such as blocking calls and requesting permissions. The frontend layer provides the user with an interactive UI by using the backend layer's methods to display and edit data. The separation of the app into 3 layers gives us the freedom to change higher layers without significantly affecting the lower layers. Together, these 3 layers provide users with a functional and easy-to-use app.*

## 2.3 System Design



# 3 DEVELOPMENT TOOLS AND STANDARDS

## 3.1 Development Tools

*The following tools are to be utilized in the development of the application:*

- *Database: RXDB*
- *Framework: React Native*
- *UI Design: Figma,*
- *iOS IDE: XCode*
- *Android IDE: Android Studio*

## 3.2 Development Standards

Tick the appropriate box to indicate the standards being followed for this application:

| Standard | √ indicates compliance |
|---|---|
| Database Design | |
| TypeScript | |
| JavaScript | |
| Accessibility | |
| Supported Android versions (7+) | |
| Supported iOS versions (10+) | |

# 4 SYSTEM PROCESSES

## 4.1 Welcome Screen

*Initial access to the application occurs through the WelcomeScreen on the front end. When the user clicks the button, all necessary roles and permissions are requested. Once the permissions are granted, the user is taken to the SettingsScreen to complete the setup process. The WelcomeScreen uses the SettingsInterface to request permissions and roles.*

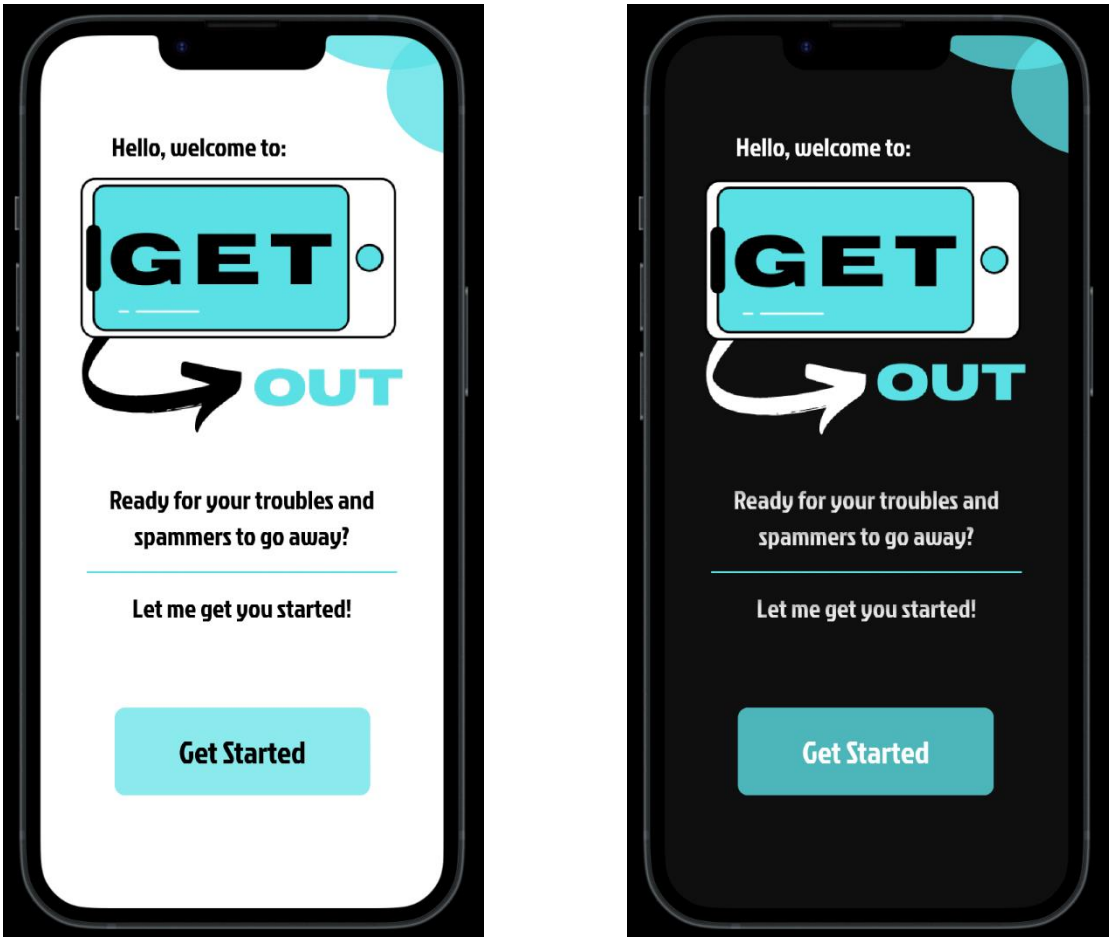



**Image 1: initial welcome screen**

## 4.2 Settings

*The SettingsScreen allows the user to change how the app works and looks. The SettingsScreen uses the SettingsInterface to fetch the current settings via the getSettings() function and update the settings via individual setter functions. Some features will require permissions separate from the ones granted during the initial setup process. For these features, the SettingsScreen will use the SettingsInterface to request permissions when the user enables them.*

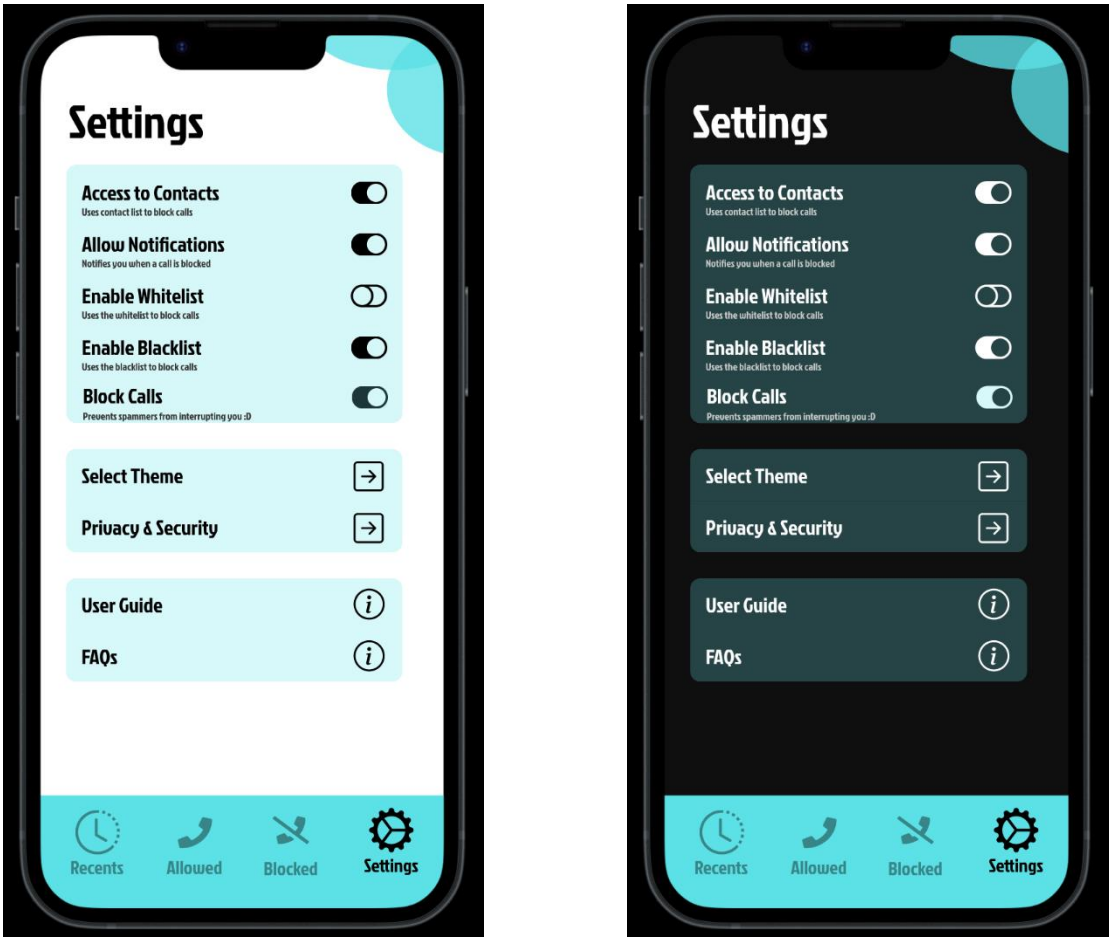



**Image 2: settings screen**

*After initial setup (or settings adjustment), the user can access each screen from the bottom navigation bar, and the main screen will be the CallLogScreen. On the backend, the application will listen for incoming calls via the CallHandler. The onScreenCall() function handles incoming calls, and it will be implemented separately for iOS and Android. It checks the phone number of the incoming call against the contact list, whitelist, and blacklist, and determines whether to block the call based on the user's settings. For the call to be blocked, it must be added manually to the blacklist from the CallLogScreen or the BlacklistScreen.*

## 4.3 Call Log

*The CallLogScreen displays a list of the user's recent calls. It uses the CallLogInterface to conveniently communicate with the database. and from the backend by the CallHandler. The CallHandler gets the metadata of the incoming call from the CallLogInterface, checking against the whitelist and blacklist to determine whether to let the call through, and then the call's metadata is inserted into the database through the CallLogInterface's insert() function. The user can utilize the front-end interface to decide whether to add a number from a received call to either the whitelist or blacklist.*

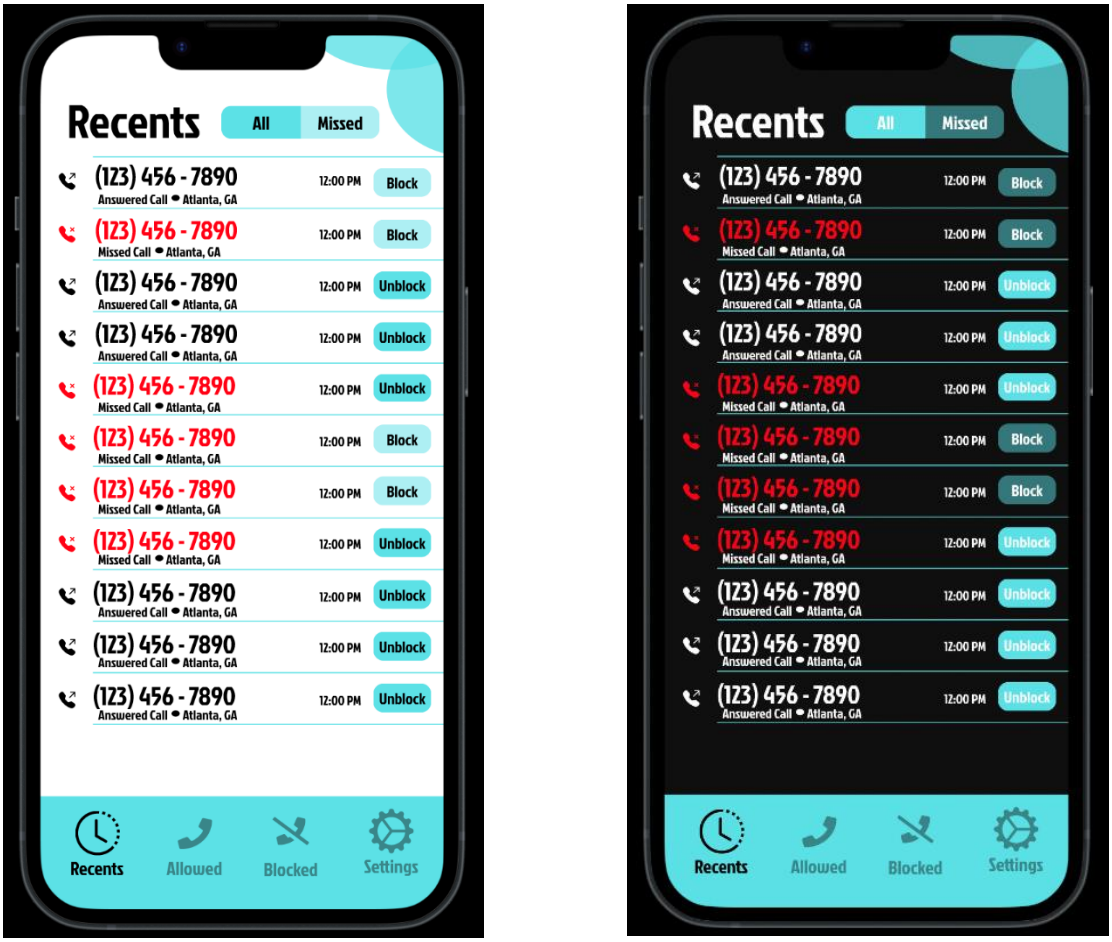



**Image 3: call log/recent calls**

## 4.4 Blacklist

*The BlacklistInterface is accessed via the CallHandler on the back end, and by the user via the BlacklistScreen on the front end. The CallHandler checks incoming phone calls against the list and determines whether to block the call based on the inclusion of part (or all) of the number on the blacklist. The user can utilize the BlacklistScreen to both view and edit the blacklist. BlacklistInterface contains the getBlacklist() function that retrieves the current blacklist from the database, and also utilizes the insert() function to update the database in response to user input.*

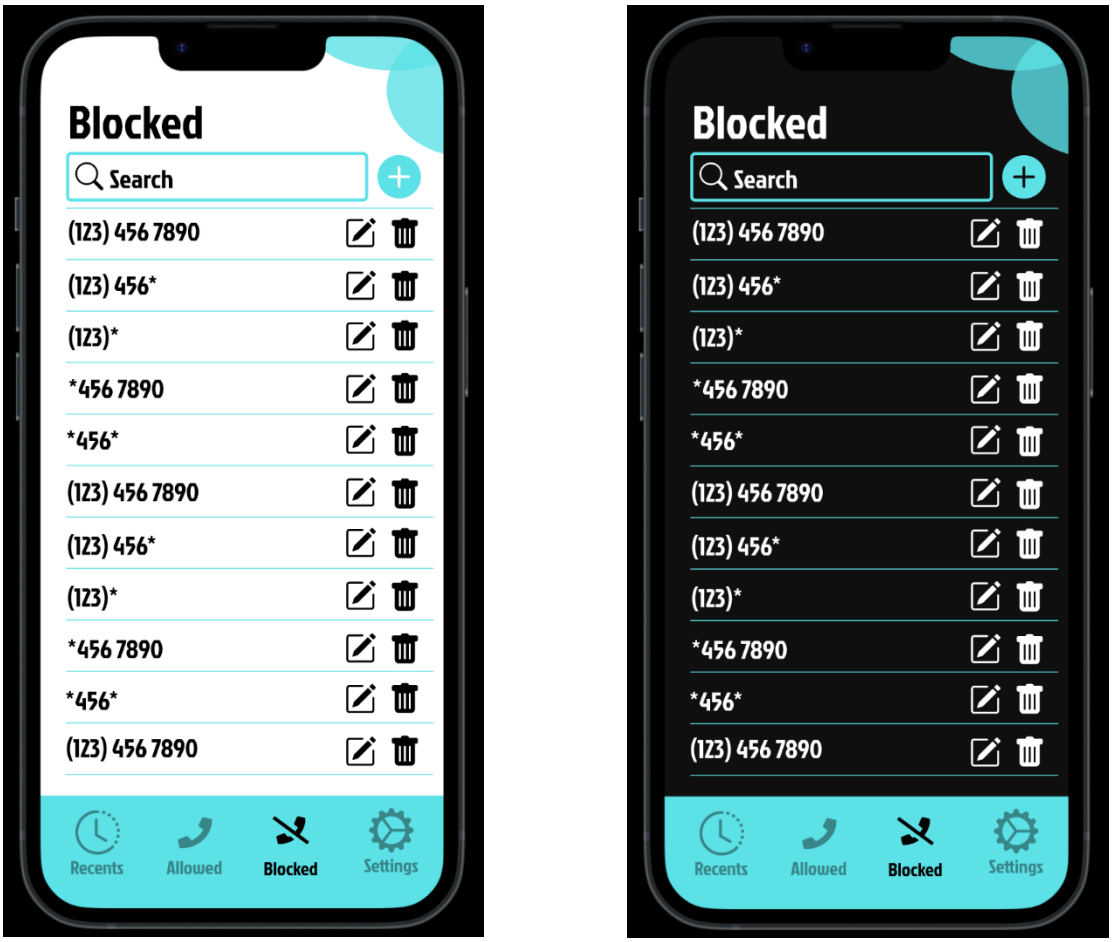



**Image 4: blacklist screen**

## 4.5 Whitelist

*The WhitelistInterface is accessed via the CallHandler on the back end, and by the user via the WhitelistScreen on the front end. The CallHandler checks incoming phone calls against the list and determines whether to allow the call based on the inclusion of part (or all) of the number on the whitelist. The user can utilize the WhitelistScreen to both view and edit the whitelist. WhitelistInterface contains the getWhitelist() function that retrieves the current whitelist from the database, and also utilizes the insert() function to update the database in response to user input.*

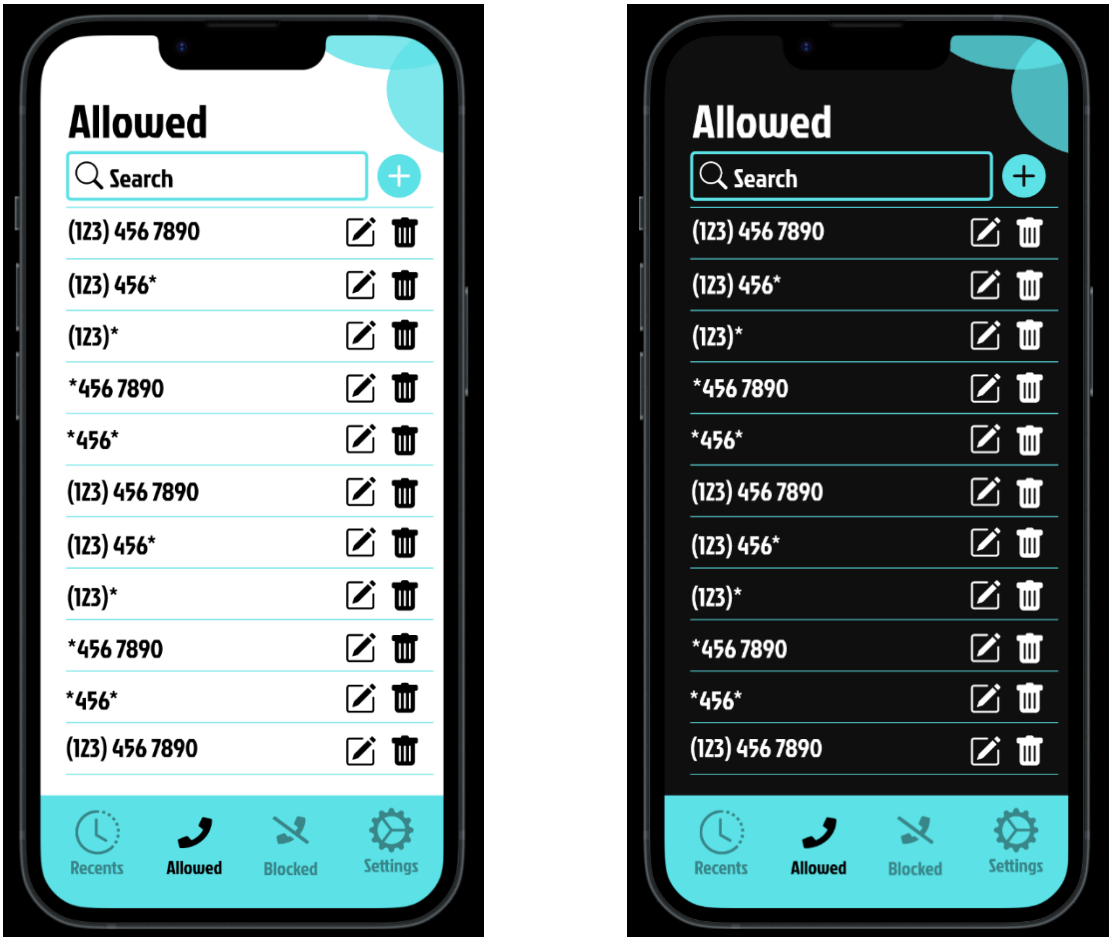



**Image 5: whitelist screen**

# 5 USER INTERFACE

## 5.1 Transactional Interface

*On a fresh installation of the app, the user will be greeted with a welcome screen, followed by a quick setup process that will fetch permissions. The setup process will only be completed once per user. Whenever the user opens the app again, they will be taken to the call log screen.*

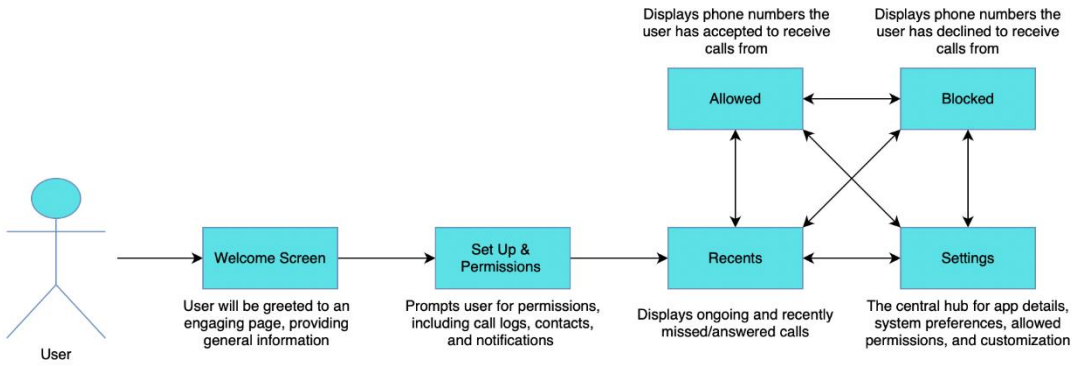



**Image 6: Use case flow**

## 5.2 Reporting Interface

*Reporting will be done via the CallLogScreen on the front end. The screen displays the user's call history, and reports whether the call was blocked by GetOut or not. Additionally, a silent notification will be displayed for the user whenever a call is blocked. When clicked, the notification will take the user to the call log screen.*

# 6 APPLICATION SECURITY

## 6.1 Authentication

*Authentication is handled by the standard Android/iOS authentication procedures. There is no separate login currently for the app.*

## 6.2 Authorisation

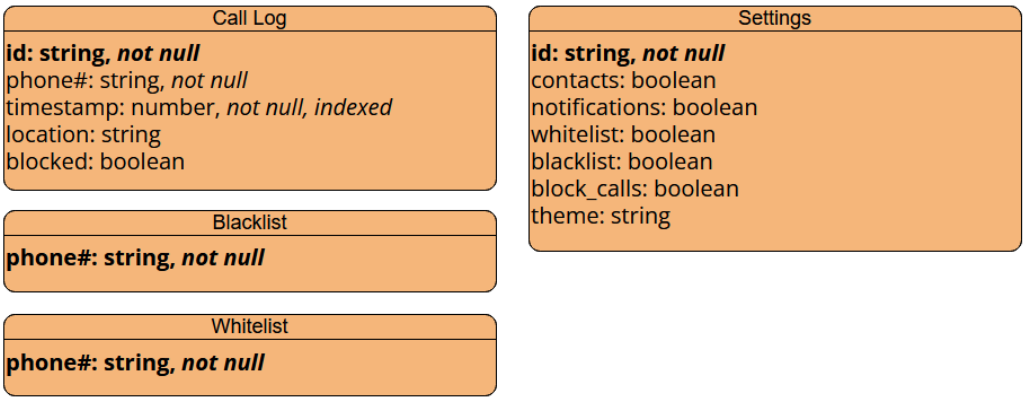
*Authorisation to access various device functions is controlled by the back end; specifically, through the SettingsInterface, which will request permission from Android/iOS devices to access, contacts, notifications, and enable white and blacklists to control call handling.*

## 6.3 Encryption

*The user's contact list and call history will be stored in the database, which could expose sensitive information. For this reason, the sensitive information will be encrypted in the database by specifying it in the RxDB schema.*

# 7 DATABASE DESIGN

## 7.1 Database ERD




## 7.2 Data Migration

*Without a database migration system, the user would have to delete their entire database to update the app. GetOut will use the following database migration system, using the built-in functionality of RxDB. With every update after the initial distribution, we will update the version number of all altered schemas and carefully specified in the schema how the data should be migrated. This will prevent the user from losing their data with each app update.*