SRS - GetOut App

1. Introduction

1.1 Purpose

The purpose of this document is to provide stakeholders and developers with an easy-to-follow roadmap of design requirements.

1.2 Intended Audience

This document is for the developers, project manager, and other faculty.

1.3 Intended Use

This SRS is intended to provide a development roadmap, as well as level set expectations for the project manager and faculty.

1.4 Scope

The scope of the GetOut app is 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. Additionally, *if time allows*, we can add spoof detection and/or 3rd-party APIs to determine if a call should be blocked, and we can add machine learning to filter spam text messages.

1.5 User Needs

Most users need a simple app that prevents unwanted callers from interrupting their day. For these users, the app needs to be easy to set up and use, which means the user interface needs to be simple and straightforward.

Other users need more advanced configuration options to specify exactly which numbers will be blocked. For these users, advanced configuration options should be accessible. However, since most users only need a simple app, the advanced options must be separate from the core of the user interface, so that it does not disrupt the user experience.

2. System Features and Requirements

2.1 Functional Requirements

- Must be able to access incoming calls, compare them to white and blacklists, and block accordingly
    - Must provide the user with the option to whitelist/blacklist incoming calls
    - Must notify the user when calls are blocked
- Must allow users the ability to access the settings menu to directly modify the black/whitelists as well as settings
- Must allow first-time users the option to customize settings on initial load
- Must have permission to access and intercept unknown incoming phone calls
- Must have permission to access contacts and determine if a phone number or email address is in the user's contacts

2.2 External Interface Requirements

User Interface

- Buttons – The user must be able to interact with the application in some way via buttons
- Text boxes – User must be presented information via text
- Images – User must be provided visual stimuli through easy-to-understand images that are directly related to functionality

Software Interface

- On first time open:
    - The order the app must load in following first time access: Splash screen – first time setup – landing pad
        - First time setup: The app will need *n* screens regarding user instructions setting up access and functionality for the app, that contain links to the settings page referenced
- Landing pad/Subsequent openings: The landing pad must display icons to access the blacklist, whitelist, call log, settings page (version, minor tweaks, assistance site), contact list
    - Settings – Must contain *n* icons/menu items that allow users to adjust whitelist, blacklist, contact list, partial number blocking and screening, and any additional settings decided on
    - Blacklist – Must contain a method to add/remove/modify numbers on the blacklist
    - Whitelist – Must contain a method to add/remove/modify numbers on the whitelist, also add them to the contacts list
    - Call log – Must display the calls received/blocked/made; must also provide the option to add a number to the contact list/whitelist/blacklist.
- Back-End Operation
    - The backend must connect to a database
    - The backend must execute SQL statements on the database
    - The backend must perform all logic necessary to handle incoming phone calls
    - The backend must expose functions that fill in dynamic elements of the front end

Hardware Interfaces

- Must be configurable to show notifications on phone screen while phone is locked
- Must be operable on mobile devices running at least Android 7+, or IOS 10+.
- Must be able to answer calls utilizing phone hardware buttons

2.3 Nonfunctional Requirements

Performance Requirements

- Must be able to run in the background constantly without draining system resources
- Must be able to quickly respond to incoming phone calls/texts
    - The decision to block calls must be made within 5 seconds of receipt of call
    - The decision to block texts must be made within 10 seconds of receipt of texts (phase 2)
- Must completely load app within 15 seconds of open
- Must download and install in a reasonably short amount of time
- Must be able to answer calls utilizing phone hardware buttons
- Must not slow down phone operation when app is minimized but still running

Security Requirements

- Must encrypt whitelist/blacklist
- Must encrypt/protect personal info input and saved in the app
- Must securely access incoming phone calls
- Must securely access the contact list on the phone
- Must securely access SMS/text messages (phase 2)