Not for terrorists secure messaging app

Table of Contents

1 Introduction

* 1. Purpose
  2. Scope
  3. Definitions
  4. Overview

...

3.3 Non-Functional Requirements

Introduction

1.1 Purpose

The purpose of this document is to explain in detail the requirements for the “Not for terrorists secure messaging app”. This document will also explain the purpose of the aforementioned system. In conclusion, this document will act as a proposal to customers as well as a reference for the development team.

1.2 Scope

The purpose of this application is to be able to send highly secure messages to those with the same app. Messages will be deleted at least 5 minutes after being read, and will automatically delete after 8 hours. Customers will be able to add encryption to their messages, as well as ten-point security patterns. The makers of this app will not have a way to view these messages, and will have minimal access to the back-end. Passwords will be only be kept on the developers servers, under standard encryption. Accounts can only be created by admins; usernames and passwords will be randomized and be at least 10 characters.

This application needs data-service or Wi-Fi to send messages.

1.3 Definitions

1.4 Overview

The document consists of a UML case diagram of all identifiable use cases, with detailed descriptions of each. Furthermore, each use case will have a corresponding sequence diagram. The next section gives a detailed UML class diagram, showing all actors, relations, etc. A brief description will also be provided.

Next will be two state machines for use cases of interest. Functional and non-functional requirements will be documented. Use case stories will be next, followed lastly by the appendix, containing minutes from the stakeholder meetings.

Use Cases

For Admins

**UC-0001 Create Account**

Creating an account for another User

Actors: Admin, System

Pre-Conditions: Admin is logged into an Administration account, user who is receiving new profile has passed company security checks.

Post-Conditions:

1. Admin clicks on ‘Create Account’ button in the toolbar
2. System generates a random 10-15 digit number ID.
3. System checks to see if number matches an existing account\*
4. System generates a random password using numbers letters and special characters
5. System checks if password matches an existing account\*
6. System returns ID and password to Admin.
7. System asks Admin to confirm creation.
8. Admin confirms
9. System creates account with generated ID and password

3a. System finds a match

3b. System generates new ID

5a. System finds a match

5b. System generates new password

User Use Cases

\*These cases are also usable by Admins

**UC-0002 Login**

User logs into their account

Actors: User, System, Admin

Pre-Conditions: User has an account; User has downloaded the app onto their mobile device

Post-Conditions:

1. User opens app
2. User types in ID and password
3. System checks account data for a match\*
4. App unlocks and shows received messages

3a. User incorrectly enters data

3b User incorrectly enters data 3 times

3c. System locks the app

3d. User must contact Admin outside of app to unlock

**UC-0003 Send Message**

User sends a message to another User

Actors: User, System

Pre-Conditions: Both Users have accounts and are logged in. Both Users must have an agreed upon encryption code and or security pattern

Post-Conditions:

1. User types in recipient ID or opens chat with desired recipient
2. User types a message into the message bar
3. User chooses time after message is read that it should be deleted.
4. User chooses whether to add encryption
5. User chooses whether to add pattern
6. User hits send button
7. System sends message to recipient ID

**UC-0004 Adding a new contacts**

Actors: User, System, Admin

Pre-Conditions: User has an account;...

Post-Conditions:

1. User selects the contact list optional
2. User types in recipient ID
3. User chooses whether to add default encryption
4. User chooses whether to add default pattern
5. User types additional notes about the recipient

2a. User types incorrect ID

**UC-0005 Open Message**

Actors: User

Pre-Conditions: The user received a message.

Post-Conditions: User opened the message.

1. User is notified that a message was received

2. User opens the app

3. User selects the unopened message

4. If the message wasn't encrypted, the user opens and reads the unencrypted message

5. Else the app will prompt the user to enter the encryption key or pattern

6. The message is decrypted and displayed

5a. Key or pattern is incorrect and the app prompts again for it

5b. Key or pattern is entered incorrectly too much and the message is deleted

**UC-006 Delete Conversation**

Actors: User

Preconditions: The user has sent or received a message or messages.

Postconditions: The conversation is deleted.

* 1. The user selects “messages”.
  2. A list of their conversations that weren’t purged already is displayed.
  3. The user selects “edit” in the toolbar at the top.
  4. The user selects the conversation(s) they want to delete.
  5. The user selects “delete”.
  6. The conversation(s) is/are deleted.

Alternative Flows:

* The user can cancel editing.
* The user can leave the Messages window.

**UC-007 Unlock Account**

Actors: Admin

Preconditions: An account was locked by the system for failed logins and the owner is verified. Also admin is logged in.

Postconditions: The account is unlocked.

* + 1. The admin selects “manage accounts” somewhere
    2. A list of accounts is displayed.
    3. The admin finds and selects the locked account.
    4. The admin selects “unlock”.
    5. The admin inputs a security code or something.
    6. The account is unlocked.

**UC-008 Delete Account**

Actors: Admin

Preconditions: An account was compromised or is inactive. Also admin is logged in.

Postconditions: The account is deleted.

* + - 1. The admin selects “manage accoutns”.
      2. A list of accounts is displayed.
      3. The admin finds and selects the account to be purged.
      4. The admin selects “delete”
      5. The admin inputs a security code or something.
      6. The account is wiped.

3.3 Non-Functional Requirements

* Security - Obviously security is a top priority for the app. Messages and accounts should be highly protected.
* Limited Accessibility - The app should only be installed on select devices, and accounts cannot be created without an admin.
* Limited Backup - Messages should not be stored on any server and should be deleted on devices after a set amount of time.
* Disaster Recovery - Should an account be hacked or a device be stolen, admins should be able to lock that account.
* Minimal Exploitability - Due to the sensitive nature of the information being exchanged, there shouldn't be any way the system can be exploited to expose that information.
* No Backend (optional) - It would be desirable to not utilize a back end for the transfer of messages.