Location Share Android Application Requirements Specification

Version 1.0

09. 24. 2013.

YeiSol Woo, Todd Welch

For ACS560 - Software Engineering Class

Instructor: Dr. Chen

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 09/20/2013 | 1.0 | First draft | Yeisol Woo |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1.0 Introduction 3](#_Toc367651535)

[1.1 Purpose 3](#_Toc367651536)

[1.2 Scope of project 3](#_Toc367651537)

[1.3 Definitions, Acronyms, and Abbreviations 3](#_Toc367651538)

[2.0 Overall Description 4](#_Toc367651539)

[2.1 System Environment 4](#_Toc367651540)

[2.2 User Characteristics 4](#_Toc367651541)

[2.3 Constraints 4](#_Toc367651542)

[3.0 Requirements Specification 5](#_Toc367651543)

[3.1 Functional Requirements 5](#_Toc367651544)

[3.1.1 General user use cases 5](#_Toc367651545)

[3.1.1.1 User logs in 5](#_Toc367651546)

[3.1.1.2 User deletes existing message 6](#_Toc367651547)

[3.1.1.3 User deletes all existing messages 7](#_Toc367651548)

[3.1.1.4 User views help documentation 8](#_Toc367651549)

[3.1.2 Sender specific use cases 9](#_Toc367651550)

[3.1.2.1 Use sends a new location message 9](#_Toc367651551)

[3.1.3 Receiver specific use cases 10](#_Toc367651552)

[3.1.3.1 User views existing message 11](#_Toc367651553)

[3.1.3.2 User views map of shared location 11](#_Toc367651554)

[3.1.3.3 User gets directions to shared location 12](#_Toc367651555)

[3.2 Non-Functional Requirements 13](#_Toc367651556)

[3.2.1 Security 13](#_Toc367651557)

[3.2.2 Data store structure 13](#_Toc367651558)

# 1.0 Introduction

## 1.1 Purpose

As people become ever connected through mobile technology, an increasingly common task is for a user to share his or her location with others. Currently this is done through sending an address or point of interest name through a text message. If an address is sent the other users must type this address into a map or navigation application to find directions. Similarly if a point of interest name is given the other users may need to search for the name in a search engine or try to find it in a mapping or navigation application.

The purpose of this application is to simplify this process by allowing a user to send his or her current location to a group of other users. Once the other users receive the message containing the latitude and longitude of the sender they will have a quick link to view the map of the location or get directions to the location.

## 1.2 Scope of project

The application will consist of (1) an Android application that will serve as the user interface, (2) a .NET server that will serve as a message relay and data access layer, and (3) a database that will act as a data store for recording user ID information.

Authentication and users’ friends lists will be handled by Facebook. Users will be able to login using their Facebook credentials and view a list of their Facebook friends that are also using the application.

The user interface will allow the user to send their location to a group of other users and to receive location messages from other users. The primary focus of the application is to share location information, no conversational messaging will be supported.

The .NET server will handle relaying messages to PUSH notification systems as well as filtering a user’s Facebook friend list to only show users that are also using the application. This list of application users will be stored in the database.

## 1.3 Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Term | Definition |
| User | End user that is accessing the system using the provided Android application user interface |
| Group | 1 or more users |
| Database | Data store that will hold the list of users and their Facebook IDs |
| User interface | Android application that allows the user to interact with the system and send their location messages to other users |
| Backend server | .NET server that will act as a message relay and data store access layer |
| Location message | A coded message that will contain a short description, the user that sent the message, the time the message was sent, and the associated latitude and longitude of the location the user was at when the message was sent |

# 2.0 Overall Description

## 2.1 System Environment

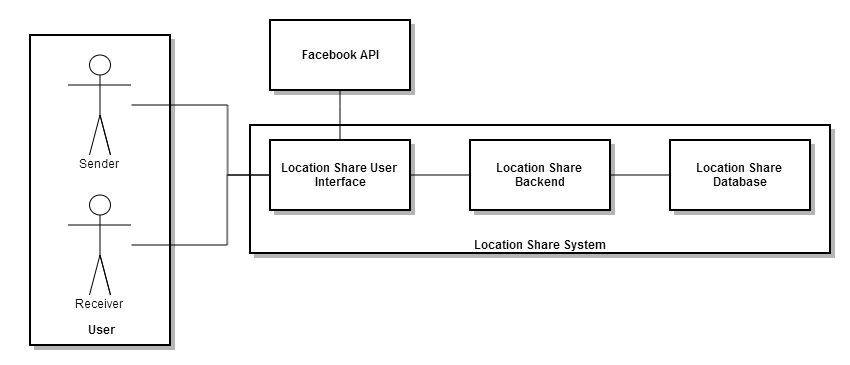


Figure - System Environment

The Location Share System has two active actors: (1) users sending location messages and (2) users receiving location messages. Each user will have a set of friends that he or she can send his or her current location to. The user will interact with the system through an Android application that connects to the backend server.

Users will be authenticated using their Facebook logins and a list of their friends will be retrieved through the Facebook API.

A list of all Location Share System users and their respective Facebook IDs will be stored in the Location Share Database.

The backend will be responsible for providing the users with a list of their Facebook friends that are also using the application. The backend will also act as a message relay, receiving the location messages from users and forwarding them to the respective receivers using PUSH notification.

## 2.2 User Characteristics

Users will have an Android 4.0+ device with support for Location Services (GPS, A-GPS, etc.), network connectivity (3G/4G or WiFi), and a Facebook login. The user will need to be able to use the Android device as well as be able to read English as well as understand simple icons.

## 2.3 Constraints

The application must consist of three parts, (1) a user interface, (2) a backend server, and (3) a database.

The user interface must run on a different operating system and be programmed in a different language as the backend server.

# 3.0 Requirements Specification

## 3.1 Functional Requirements

### 3.1.1 General user use cases

**Diagram:**

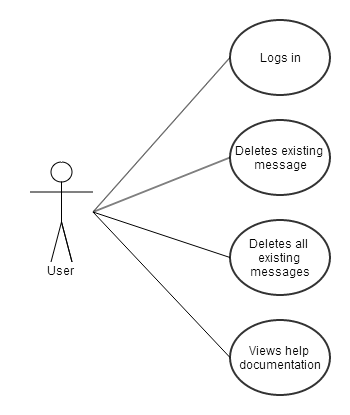


Figure - General user use cases

### 3.1.1.1 User logs in

**Diagram:**

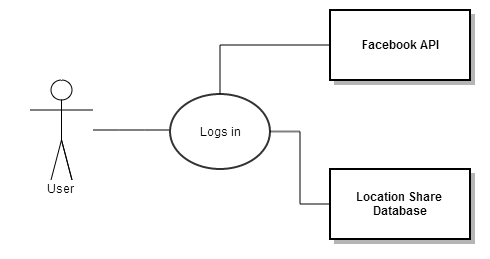


Figure - User logs in use case diagram

**Brief description:**

The user logs into the system by providing their Facebook credentials which are verified using the Facebook API.

**Preconditions:**

Before the user can log in they must install the Location Share application on his or her compatible Android device and launch the application. The user must not already be logged into the system.

**Step by Step Directions**

1. The application verifies the user is not already logged into the system.
2. The application presents the user with the standard Facebook login widget (provided by Facebook SDK).
3. The user types in their username and password that is tied to a valid Facebook account.
4. The Facebook API authenticates this username and password against Facebook internal systems.
5. If the authentication was successful then the application notifies the server and the server stores the user’s Facebook UID in the database. The user is then shown the user interface for viewing / sending location messages.
6. If the authentication is not successful then the login widget displays an error message and the user tries again (step #3).

### 3.1.1.2 User deletes existing message

**Diagram:**

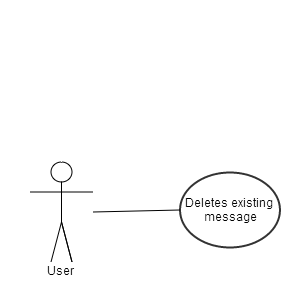
****

Figure - User deletes existing message use case diagram

**Brief description:**

The user deletes a single existing message from the application. The message is not deleted in another users’ application, just this user’s application.

**Preconditions:**

The user has launched the application and is on the home screen.

**Step by Step Directions**

1. The user clicks the application’s menu button.
2. The application displays a context menu with available actions.
3. The user clicks the ‘View Existing Messages’ button.
4. The application hides the context menu and displays a dialog displaying existing message.
5. The user selects a single message.
6. The application displays a context menu with available actions.
7. The user clicks the ‘Delete’ button.
8. The application deletes the location message, removes it from the map, hides the context menu, and displays the updated list of existing messages. The user can click the ‘X’ in the dialog to return to the home screen.

OR

1. The user clicks the icon for a message on the map.
2. The application displays a context menu with available actions.
3. The user clicks the ‘Delete’ button.
4. The application deletes the location message, removes it from the map, hides the context menu, and displays the updated map.

### 3.1.1.3 User deletes all existing messages

**Diagram:**

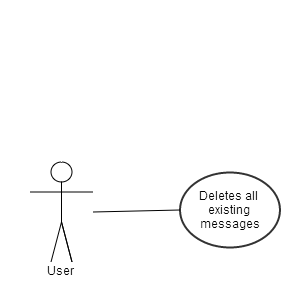
****

Figure - User deletes all existing messages use case diagram

**Brief description:**

The user deletes all existing messages form the application. These messages are not deleted from other users’ application, just this user’s application.

**Preconditions:**

The user has launched the application and is on the home screen.

**Step by Step Directions**

1. The user clicks the application’s menu button.
2. The application displays a context menu with available actions.
3. The user clicks the ‘Delete Existing Message’ button.
4. The application displays a confirmation dialog.
5. If the user clicks ‘OK’ then all message are deleted, the confirmation dialog and the context menu are hidden, and the updated map is displayed.
6. If the user clicks ‘Cancel’ then the confirmation dialog is hidden. The user can click the application’s menu button again to hide the context menu.

### 3.1.1.4 User views help documentation

**Diagram:**

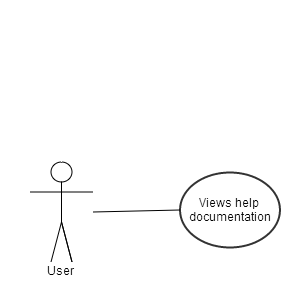
****

Figure - User views help documentation use case diagram

**Brief description:**

The user can view a brief how-to for assistance in using the application.

**Preconditions:**

The user must have the application launched, on the main screen.

**Step by Step Directions**

1. The user clicks the menu button in the application.
2. The application displays a context menu with available actions.
3. The use clicks the ‘Help’ button in the context menu.
4. The application hides the context menu and displays a popup containing a scrollable HTML viewer that shows the user how to complete basic tasks in the application.
5. The user can hide the help window by clicking the ‘X’ in the window.

### 3.1.2 Sender specific use cases

**Diagram:**

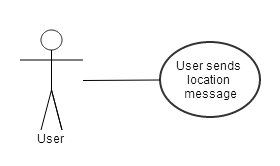
****

Figure - Sender specific use cases

### 3.1.2.1 User sends a new location message

**Diagram:**

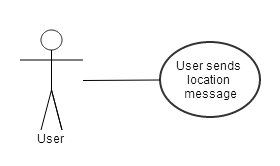


Figure - User sends location message use case

**Brief description:**

The user can send a location message to a group of one or more users that contains their current location, which allows the receivers to use this location to view it on a map or get directions.

**Preconditions:**

The user must have the application launched, on the main screen.

**Step by Step Directions**

1. The user clicks the menu button in the application.
2. The application displays a context menu with available actions.
3. The user clicks the ‘Send Location Message’ button in the context menu.
4. If the user wants to cancel the send location message action he or she can click the ‘X’ on the dialog. The application will then hide the dialog and show the context menu.
5. The application displays a dialog asking the user for a short description of their location.
6. The user enters a short description and clicks the ‘Next’ button.
7. The application displays a list of available friends to send the location message to.
8. The user checks the check box next to each available friend he or she wants to send the location message to and clicks the ‘OK’ button.
9. The application sends the location message to the server.
10. The server forwards the message to the PUSH notification system for distribution.
11. The receivers get a PUSH notification.
12. The application hides the dialog and context menu and displays the map.

### 3.1.3 Receiver specific use cases

**Diagram:**

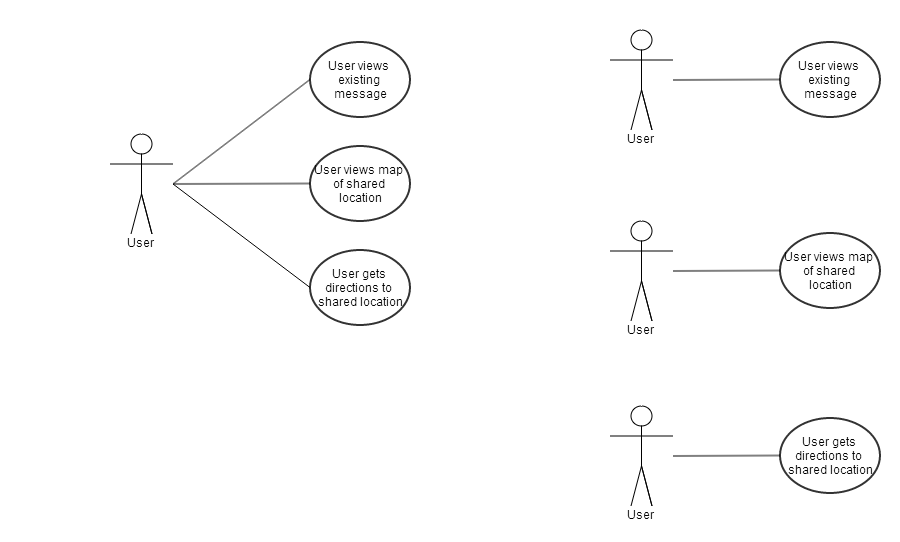


Figure - Receiver specific use cases

### 3.1.3.1 User views existing message

**Diagram:**

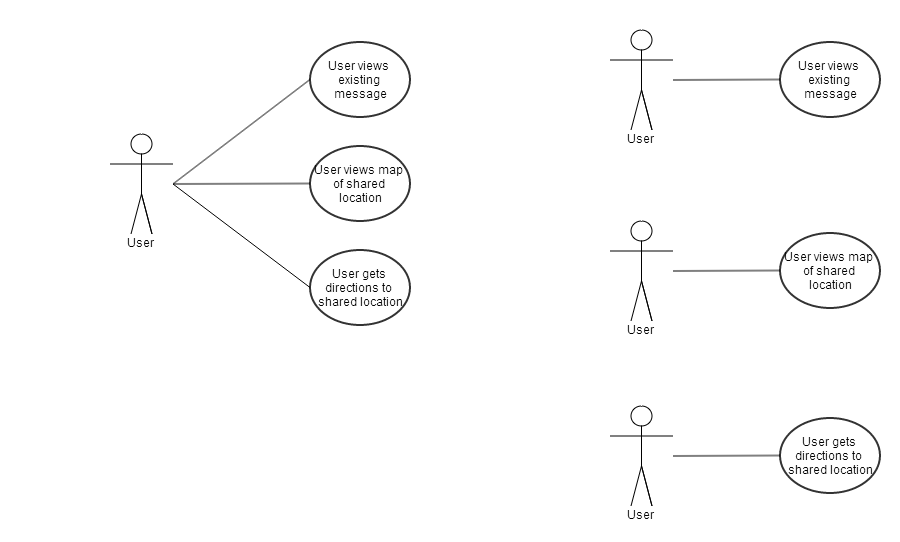


Figure - User views existing message use case

**Brief description:**

The user can view existing message (either received by the user or sent from the user) along with a context menu of available actions.

**Preconditions:**

The user must have the application launched, on the main screen.

**Step by Step Directions**

1. The user clicks the show existing messages button.
2. The application displays a list of existing location messages.
3. The user scrolls through the list and selects a location message.
4. The application displays the message in the message viewer dialog along with a context menu of available actions.
5. The user can click the ‘X’ button to hide the message viewer dialog. The application will then display the existing messages list.
6. The user can click on the different action buttons such as ‘Delete’, ‘View Map’, or ‘Get Directions’ to perform various actions with the location.
7. The user can click the ‘X’ button in the message list to hide it.

### 3.1.3.2 User views map of shared location

**Diagram:**

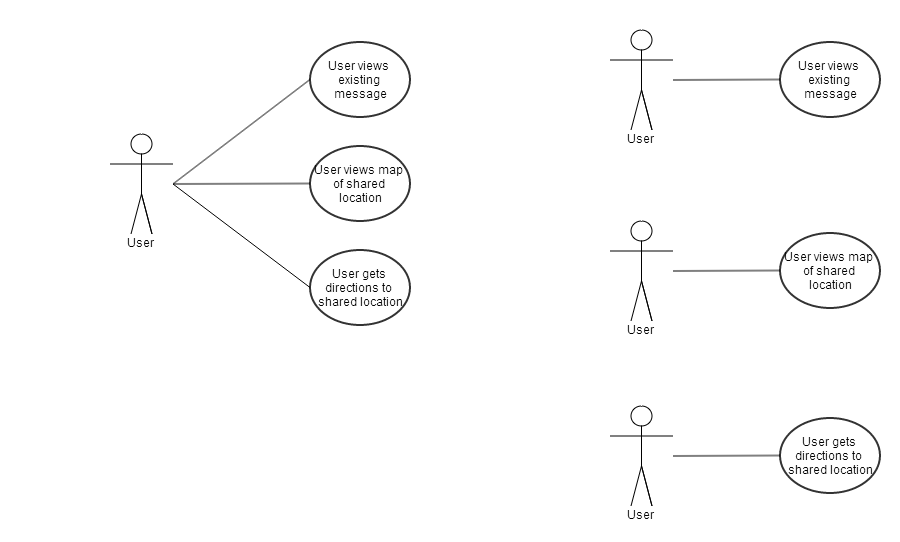


Figure - User views map of shared location use case

**Brief description:**

The user can view a map of the shared location and its surrounding area to get a better idea of where the sender is at.

**Preconditions:**

The user must have the application launched, has selected a specific location message, and is viewing the message in the message viewer dialog.

**Step by Step Directions**

1. The application displays a context menu in the message viewer dialog.
2. The user clicks the ‘View Map’ button in the context menu.
3. The application hides the message dialog and the context menu.
4. The application uses the Mapping Services API from the Android device to display a map of the shared location including a given radius of the surrounding area.

### 3.1.3.3 User gets directions to shared location

**Diagram:**

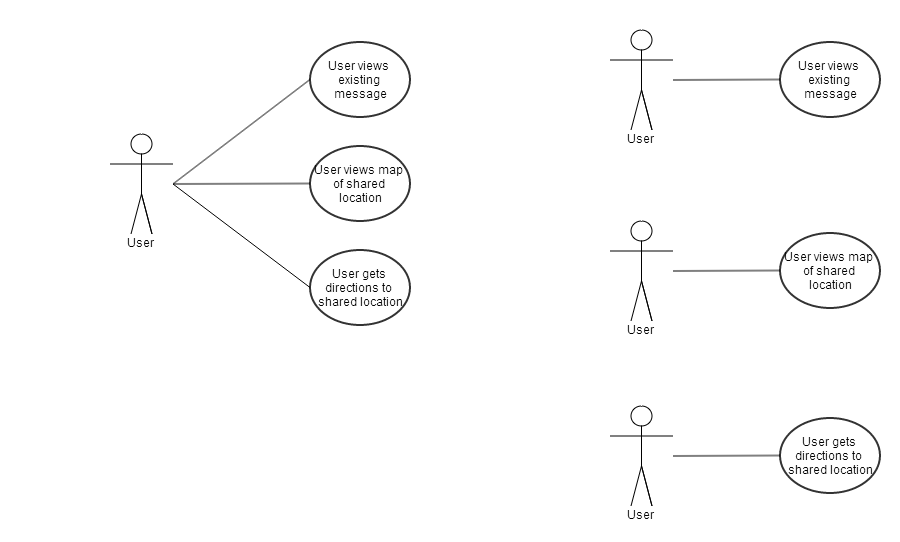


Figure - User gets directions to shared location use case diagram

**Brief description:**

The user can get navigation directions to the shared location from their current location using the Map Services API on their Android device.

**Preconditions:**

The user must have the application launched, has selected a specific location message, and is viewing the message in the message viewer dialog.

**Step by Step Directions**

1. The application displays a context menu in the message viewer dialog.
2. The user clicks the ‘View Map’ button in the context menu.
3. The application hides the message dialog and the context menu.
4. The application uses the Location Services API from the Android device to get the receivers current location.
5. The application forwards this information along with the destination location from the location message to the Mapping Services API from the Android device to display the navigation application (which then displays the necessary directions to get to the shared location).

## 3.2 Non-Functional Requirements

### 3.2.1 Security

Authentication will be proxied by the backend server and handled by Facebook by allowing a user to login with their Facebook IDs. Friends lists (which act as an ACL for messaging) will be refreshed every time the user attempts to send a new location message.

Local caching of Facebook pictures and messages will be done on the Android device. No messages will be stored on the backend server or database, they will only be forwarded to the PUSH notification system.

### 3.2.2 Data store structure

The Location Share database will consist of a single table which lists all users of the system and their associated Facebook ID. This table will assist in filtering a user’s Facebook friends list to only display other users of the application.

|  |  |  |  |
| --- | --- | --- | --- |
| Data Item | Data Type | Description | Comment |
| FacebookID | VARCHAR | Variable length (max 128) unique ID for Facebook user | Provided by Facebook |
| FirstName | NVARCHAR | Variable length (max 128) user’s first name | Provided by Facebook |
| LastName | NVARCHAR | Variable length (max 128) user’s last name | Provided by Facebook |
| DisplayName | NVARCHAR | Variable length (max 256) user’s Facebook display name | Provided by Facebook |
| LastLogin | DATETIME | Last time user logged in to the system | Used to delete expired records |