1 Introduction

* 1. Background

## Assumption

1. Solution Overview
   1. Implementation
      1. Design Flow
         1. Configuration Management

The mobile app should fetch the static configurations from the server whenever main activity is loaded provided the older configuration is more than 4 hours old.

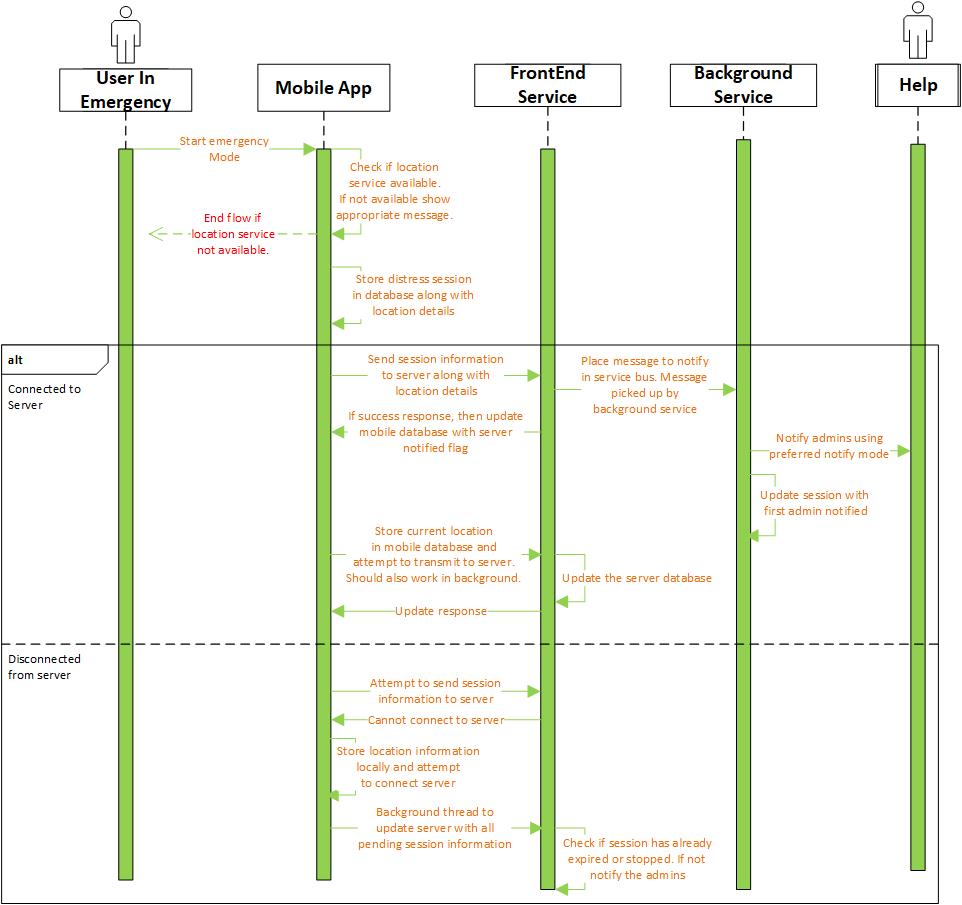
* + - 1. Group Management

User can create groups. While creation there can be an option to make the group public. User can edit a group to make the group public/private. Public groups can be searched by others for taking help from the public group. If a person makes a group as private, the existing shared data (eg. Shared emergency sessions) will still be visible to the group administrator, however, the group will not be visible to the public. To make a public group or to make a group public, the name of the group should be unique amongst all the public groups. A filtered unique constraint can be defined on the database for the same. The creation or edit of a group should occur in connected mode on server.

The group can have one or more administrators who have following access:

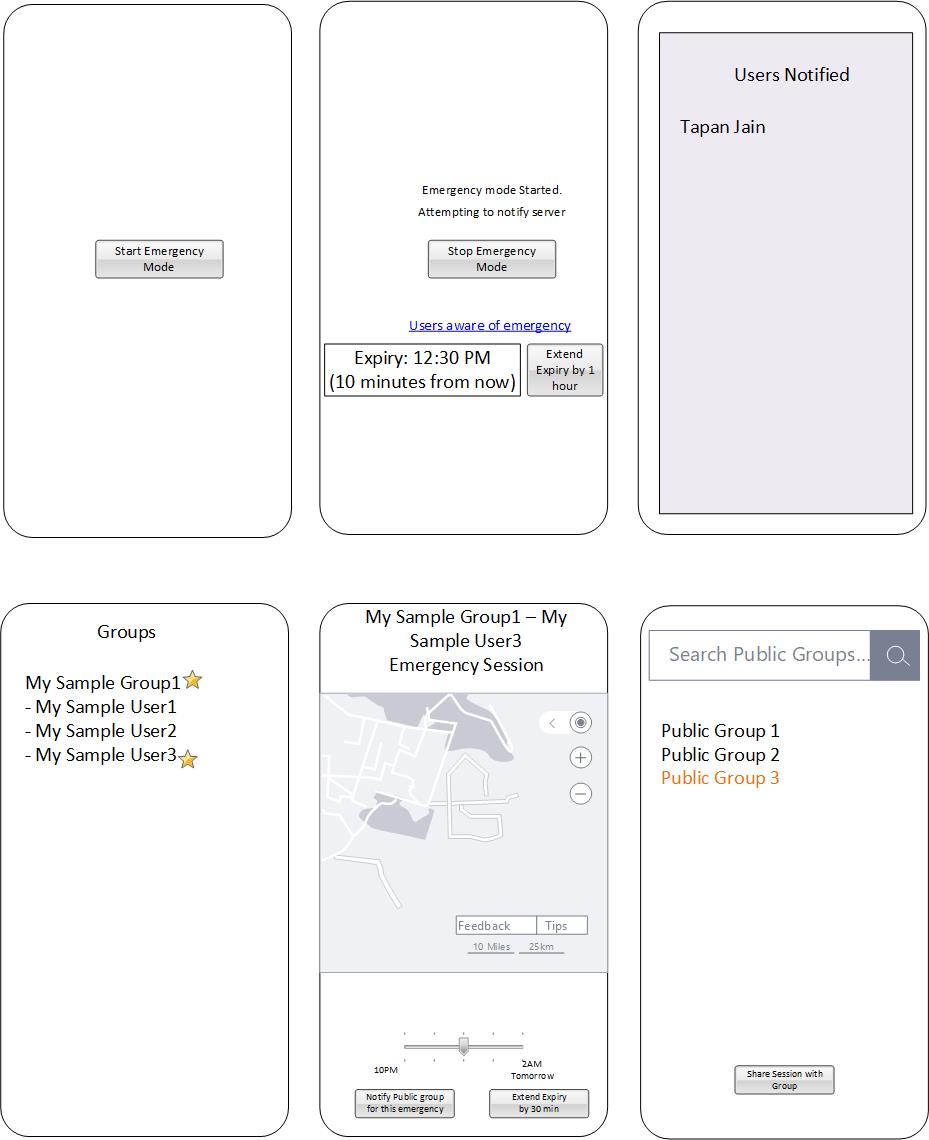
1. User’s location information during an emergency session.
2. Request user’s location.
3. During an emergency session, the user in emergency cannot be removed as there is security risk of losing user data. However, the administrators can remove other administrators or users to avoid the data being read by other users. Additionally, other administrators can be added like police officers to view the location. The administrators can stop the emergency session in case it was triggered by mistake by user and user does not have access to his/her mobile. If the administrator stops the emergency session, then the administrators id should be used in the stopped by column. There should not be any foreign key relationship in database as later the user may be deleted from system because of user requesting removal.
   * + 1. Emergency Mode

This mode is used whenever there is an emergency that needs immediate attention. This helps notify all the administrators of the primary groups or a group of user’s choice.



1. User initiates the emergency mode by clicking the Emergency button. On clicking the emergency button, the user should be displayed a custom message that everyone can view. There should be an option to edit the existing message or pick a message from the available templates. If the location is not available or enabled, user should be displayed the appropriate message and emergency mode should not start. Whenever user initiates an emergency mode, the app should store the emergency session along with the current location in the local application database. The current location is needed with every emergency session trigger request to server.
2. Thereafter trigger a request to server to initiate the emergency mode. If the request cannot be sent to the server due to data connection issue, the user should be displayed an appropriate message.
3. During an emergency session, below items should be visible on the main screen corresponding to a session
   1. Stop emergency button
   2. Link to view the list of users who have viewed the emergency session. The list of users who viewed emergency session should be fetched with every call made to update the location in the emergency mode.
   3. Expiry time for the current session.
   4. Button to “Extend Session expiry”.
   5. Status of the emergency session. Should be displayed as below:
4. Emergency mode started. Attempting to notify server.
5. Server notified, last location sent at 12:22 PM (30 seconds ago or 2 minutes ago)
6. Admin notified, last location sent at 12:22 PM (30 seconds ago or 2 minutes ago)
7. Emergency mode started. Data connection is not available now. Others will be notified once the data connection is available.
8. Emergency mode started. Issues in connecting to server. Others will be notified once connection is established.
9. During emergency mode, the location capture and transmission should happen every 1 minute (configurable value between 30 seconds to 5 minutes). While triggering an emergency session, the primary group should have access to the emergency session. While creating an emergency session, user can optionally specify the groups to which the emergency session should be accessible. Here the user can specify the public groups as well. The administrators of the groups who have access to emergency session should get a push notification and/or email whenever an emergency session is received on the server based on the preferred notification mode. The email should have the location of the user where emergency was reported.
10. After the administrator receiving the emergency push notification clicks the push notification, he/she would be taken to the page where the recipient can track the location of the user in emergency. Optionally the recipient can go to the app and select the group that is highlighted to have a person in emergency and then go inside and check the person in emergency and then view the emergency sessions and navigate to the active emergency session. Active emergency session is a session that has not yet been closed and lies within the expiry period of the session.
11. The expiry period of a session is set at the start of the session based on the configurable timeout period (*EmergencySessionTimeoutInSeconds*). The expiry for an emergency session can be extended by the user or the administrator of the group having access to emergency, provided that the user has given access to the group administrators to perform the extension. By default, it is enabled. The option in mobile should be there to increase the time of capture as “extend capture time by 30min/1 hour/3 hour/6 hours/12 hours” and associated price should be shown. If the user has extended the session beyond the free limit the user would be charged on his/her account. If the user does not have a card associated with his/her account or if the payment bounces then the user’s account would be put on hold after a period of 7 days since the latest negative balance on the account. Thereafter, the user will not be able to perform basic actions on the account. To avoid misuse of this feature, the maximum limit till which the session can be extended is 12 hours (configurable value). Post this, the user should contact the call center to make the change. As the battery would be utilized, the emergency mode should not run for a long-time else battery for making the call would not be available.
12. The group administrator should be able to trace the history user locations along with the time at which the user was present at any location. There should be a timeline available to view the location of user at specific times. The timeline should have the start time and end time of the emergency session and it should be scrollable up to the point till which the emergency data is available. The mobile app should store the count of times the location has not changed during an emergency session at a location. Based on the number of repeat counts, the map should show the polyline between two points in either green, yellow, orange or red (green for 1-3, yellow for 4-10, orange for 11-20, red for above 20). The speed of the person in emergency should also be available in case the administrator is trying to identify the mode of transport for the person in emergency to identify the vehicles to be checked to reach the person in emergency. By sliding the timeline, the user would be able to view the points on the map highlighted showing time and speed. Similarly, clicking the location points on the map should adjust the timeline. The emergency data should be available for future reference for a period of 7 days.
13. Facility should be available to get snapshot of the emergency consisting of location, speed and time information (using timeline or clicking at a location point) to get either direction to the snapshot or to send the snapshot to someone.
14. User can make use of a Bluetooth low energy device to send signal to mobile to start the emergency mode. There should be a wide variety of devices at reasonable price so that anyone can purchase it.
15. A user in emergency or a administrator of a group can share an emergency session information with a public group. The sharing will allow the selected public group administrators to have access to the emergency session.

**Mobile Screens**



Caveats:

1. During group creation make sure that you add the members in the group whom you trust as they can monitor your location.

Enhancements: 1) The notification regarding emergency can be sent to the dedicated person/group responsible for security in the region if a verified security group is added by default.

1. Police/Army can be associated with a region for securing the public and each region would have a dedicated security personnel assigned.
2. During an emergency session a person can take audio/video samples and submit them to the server. The audio/video samples are encrypted and stored on company server. The decryption of the content should lie with another department or another company that should specialize in the tackling of cases. They would be accessible only after verification that it is not being misused.

Misuse Prevention

1. Each group cannot have more than 5 members in the free subscription. As the free subscription may be used by people for tracking beggars, prostitutes at large scale where people are forced to do an activity. Whenever anyone purchases a subscription, the reason for purchase and customer visit should be performed to validate the authenticity of the customer. Still, illegal people can make use of the app by managing 1 agent per 5 subordinates. Models should be designed to identify repetitive usage patterns collaborated with the areas where the monitoring is happening to determine if it is being used for any illegal purpose.

Misuse by Party causing emergency

1. The assailant can make use of the app to call the security/help at a specific location where he/she has planned an attack on the help. So, before a dedicated personell/team is assigned for a location, ascertain that the personnel would be secure at the location where help is needed.

CounterMeasure: A caveat should be shown on the members page regarding members of the group being able to track the location of the user and add only the users you trust.

1. Gangs of robbers or thieves or gangsters can make use of the tool to call for help or to track their members during an immoral activity.

CounterMeasure: The activity from these gangs can be associated with any mishappening around an area and people can be tagged as suspicious based on this. This would be helpful in determining the next activity from such people which can be monitored.

1. People can force a girl and take video/audio of the girl and upload it on the app that can be shared with others.

CounterMeasure: The video/audio can be uploaded to the server in encrypted format and also kept on the mobile in encrypted format. This video/audio will not be accessible to public. Only in case of a registered case can this data be revealed to the police or other security authorities.

1. Public can create fake audio/video samples and create a fake session and post the video on that fake session by encrypting a fabricated video using the same technology that is present in the mobile app. So, the video/audio recording present on server cannot be considered as a proof of any case as it may be fabricated. However, it can still be analysed by the court to determine if it is fabricated. All the app is providing is a feature of capturing the audio/video during an emergency. So, instead of concentrating on capturing the complete audio/video, during an emergency session the app may determine every 5 minutes within an emergency session if any audio can be taken or video can be taken based on whether any audio input is coming or if any light is present in front of camera. Need to check if automatic capturing of video/audio (though after the user sets this in the settings) is legal.

Applicable use cases

1. Kidnapping/Abduction: While a person has been kidnapped and the person needs help.
2. Stuck during an attack by bank robbers/burglary/terrorists etc.
3. Moving out of the house during threat from extremists or during curfew.
4. This can be used for handling repetitive harassment in which case the victim can create an emergency session and take the audio/video of the event. There should be a separate company that would be responsible for analysing the audio/video content uploaded by the user and responsible for misuse of data. Each decoding should be charged. Every user should be given a specific number of free decoding and post the threshold the user would be charged based on number of requests. Once the content has been verified and provided to the victim, the victim can lodge a complaint. Every request to decode should have a valid reason for the request and should be categorized under a list of valid scenarios.

**Server Database**:

Every data coming from mobile that can occur in disconnected mode should have a request time that should be added to server database to keep track of when the request was received vs when the data reached the server. Eg. The emergency mode can happen in background.

1. User
   1. UserId
   2. PhoneNumber (Unique)
   3. Email (for recovery of password, or to change the account phone number)
   4. EncryptedPassword
2. Device
   1. DeviceId
   2. SupplierId
   3. DeviceCode – 12 digit number, first 3 digit specify the supplier, last 9 digit specifies the device unique number for the supplier. Can support max 999 million devices per supplier.
   4. DeviceType (GPS Device, Camera Device etc.)
   5. DeviceRegistrationCodeEncrypted – This is a 6 digit alphanumeric string that can be used by the user for first time registration. This would be supplied in a separate paper while delivering the device to the consumer. It is AES encrypted.
3. UserDevice
   1. DeviceId
   2. UserId
   3. DeviceFriendlyName – Name given by the user to the device he owns. Should be unique among the devices owned by the user for easy identification of the device. When a user creates a group, user can add the devices based on friendly name of the devices (or view all devices owned by user) or by using the DeviceCode. If some other user is trying to add a device not belonging to the user to the group, that user needs to use the complete devicecode to add the device to the group.
4. EmergencySession
   1. EmergencySessionId
   2. Title<Emergency\_UserTriggered\_20180726063121333>(Unique constraint)
   3. ExpiryDateTime
   4. firstAdminNotified
   5. firstAdminNotificationDateTime
   6. EmergencyTargetUser
   7. IsEmergencyRequestInProgress (This flag would be set when an admin has requested the creation of an emergency session on the target user and the user’s mobile has not yet synched. While an emergency session request is in progress and not expired, no other session request can be made)
   8. CreatedBy
   9. RequestDateTime(This should come from mobile app)
   10. CreationDateTime
   11. StoppedBy
   12. StopDateTime
5. EmergencySessionExtension
   1. EmergencySessionId
   2. RequestDateTime(Used only when the user in emergency requests the extension. Administrator initiated extensions should happen while admin is connected to server)
   3. IsExtensionRequestInProgress (This flag would be set when the admin has requested extension and the mobile app has not yet increased the end time. During this mode no other extensions can be performed)
   4. CreatedBy
   5. CreatedDateTime
6. EmergencySessionViewer
   1. EmergencySessionId
   2. GroupUserId
7. EmergencySessionGroupAccess
   1. EmergencySessionId
   2. GroupId
8. EmergencyLocation
   1. LatitudeEncrypted
   2. LongitudeEncrypted
   3. SpeedEncrypted
   4. SameLocationRepeatCount
   5. EmergencySessionId
9. Group
   1. GroupId
   2. GroupName
   3. GroupCategoryId
   4. IsPublic (Public groups can be searched by other people for sharing information)
10. GroupCategory
    1. GroupCategoryId
    2. CategoryName (Personal, Security organization, Public Health)
    3. CategoryDescription
11. PublicGroup
    1. PublicGroupId
    2. GroupId
    3. IsVerified
    4. VerifiedTitle – The verified title should be effective in user search. (It should clearly have the state/city/country/locality being served for easy search by user.)
    5. VerifiedDescription
12. CountryBasedPublicGroup - these options would help user select the providers based on nation/state/city/locality. The groups would not be auto-filtered when user searches in the search bar based on user’s location. User has to explicitly specify the filters to get filtered results. Additionally, the public groups during creation should specify the list of city/state/locality that they support.
    1. CountryId - The public groups would be visible for all to search, so the data can be limited based on the country of the user. The mobile app in every country should have a unique identifier. Based on the identifier, specific groups can be returned.
    2. PublicGroupId
13. StateBasedPublicGroup
    1. StateId
    2. PublicGroupId
14. CityBasedPublicGroup
    1. CityId
    2. PublicGroupId
15. LocalityBasedPublicGroup
    1. LocalityId – The distinct localities can be fetched based on the pin code excel downloaded from the India government site.
    2. PublicGroupId
16. EmergencySessionPublicGroupAccess
    1. EmergencySessionId
    2. GroupId
17. GroupMember
    1. GroupMemberId
    2. GroupId
    3. UserId
    4. IsAdministratorAllowedToTriggerEmergencySession
    5. IsAdministratorAllowedToExtendEmergencySession
    6. EmergecyNotificationModePreference(Sms, Email or both- default both)
    7. IsAdmin
    8. IsPrimary – By default the first group created by the user becomes the primary group
18. GroupDevice
    1. GroupDeviceId
    2. GroupId
    3. DeviceId
    4. IsAdministratorAllowedToExtendEmergencySession
    5. IsAdministratorAllowedToTriggerEmergencySession

**Mobile Database**

1. EmergencySession
   1. EmergencySessionClientId
   2. EmergencySessionId (This should be set for an emergency session that is initiated from the server by an administrator or when an emergency triggered by the user has been notified to the server.)
   3. Title
   4. ExpiryDateTime
   5. firstAdminNotified
   6. firstAdminNotificationDateTime
   7. RequestedBy
   8. RequestDateTime
   9. StoppedBy
   10. StopDateTime
   11. IsSessionStartNotifiedToServer
   12. IsSessionStopNotifiedToServer
2. EmergencySessionExtension
   1. EmergencySessionExtensionClientId
   2. EmergencySessionExtensionId(This should be set for an extension that is coming from server or an extension that is notified to the server.)
   3. EmergencySessionClientId
   4. RequestedBy (This will be set if the extension request is made by the user in emergency)
   5. RequestDateTime
   6. NewExpiryDateTime
   7. ExtensionRequestStatus (None, InProgress, Applied, ReceivedPostStop, LessThanExpiry. This should be set based on whether the extension is applied, or if the extension is received post the expiry or stopping of the session, or if the extension requested is less than the expiry which means user would have already extended the session. For an extension requested by user it should be none)
   8. IsServerNotified (Server needs to be notified regarding a new extension created by the person in emergency. Also, it needs to be notified when a extension request that comes from an administrator is applied)
3. Location
   1. EmergencySessionId
   2. Latitude
   3. Longitude
   4. Speed
   5. RepeatCount
   6. IsServerNotified

**Configuration**

1. EmergencySessionTimeoutInSeconds (Can be set by user as 30 min to 6 hours)
2. MaxEmergencySessionTimeoutforFreeSubscriptionInSeconds (6 hours, comes from server)
3. EmergencySessionAvailabilityPeriodInSeconds (7 days, higher for paid version, used in server only)
4. MaxEmergencySessionTimeoutFromAppInSeconds (comes from server, can be set as 12 hours)
5. MaxWaitTimeForPendingPaymentOnAccount (stored on server, can be set as 7 days)
6. MaxEmergencySessionLocationNotificationBatchSize (stored on client, can be set as 20)

**API’s**

// For an emergency session, the mobile app will maintain a EmergencyClientSessionId(This is a surrogate key stored in mobile to uniquely identify session record till the EmergencySessionId is not available). The data from the mobile app will be cleared once the server has been notified to stop emergency session. On every stop emergency session, mobile app will notify stop for current and then the stop for all other remaining sessions pending stop notification. After each stop is notified the records would be deleted from the mobile app. There should be a background thread that should run every 1 hour and check if there is any pending emergency session to be notified in the app. Additionally, when the user opens the app, the background thread should be triggered if not already running. On the server the emergency session name should be unique in case multiple records are created.

1. Start emergency session

SessionStartResponsModel StartEmergencySession(SessionStartRequestModel)

SessionStartRequestModel(Title, ExpiryDateTime, RequestDateTime)

SessionStartResponseModel (EmergencySessionId)

1. Extend session expiry

// Below API is used when there is a request to extend the session time by the user in emergency

ExtendSessionExpiry(EmergencySessionId, RequestDateTime, RequestedBy, NewExpiryDateTime)

// If EmergencySessionId is not available to the mobile for an actively running session on mobile

//then every time a location update is requested, we update the mobile database and call the

// StartEmergencySession API. Once the startEmergencySessionAPI completes, the API for // extendSessionExpiry is called for all session extension requests in local database marking them as ServerNotified and thereafter the call for bulk update of location request should be sent as below in batch of 50 requests. If no more data is available, the stop call is made if the session has already stopped.

1. Update Emergency session with location

// For below API the EmergencySessionId should be available to client and should always occur post update request

SessionLocationUpdateResponseModel UpdateEmergencySession(SessionLocationUpdateRequestModel)

SessionLocationUpdateRequestModel (EmergencySessionId, latitude, longitude, speed)

SessionLocationUpdateResponseModel (EmergencySessionId, ExpiryExtensionModel, StopRequestModel, list of users already viewed the session, firstAdminNotified, firstAdminNotificationDateTime)

ExpiryExtensionModel(NewExpirationDateTime, RequestedBy, RequestDateTime) – If this model is null, then expiry is not requested, else expiry is requested.

StopRequestModel (RequestedBy, RequestDateTime) – If this model is not null then stop is requested.

1. Stop an emergency session

StopEmegencySession(EmergencySessionId, RequestTime, RequestedBy (System in case of session time expired), StopReasonId)

StopReasonId(SessionTimeExpired, StopRequested)

1. Update Emergency Session bulk API

UpdateEmergencySession(EmergencySessionId, IList<SessionLocationUpdateRequestModel>)

* + - 1. Get Current Location for another user in group
      2. Share the location with others in group
      3. Share the location with others using messaging media on mobile
      4. Notification on entering or exiting a geo-fence