Project Report

SafeDrop

TEAM # 9

Sankha Pathak
Pooja Desai
Xuejiao(Donna) Tang

1. Produce Overview

1.1 Description

SafeDrop is an application both for people who need safe night pickup service and proactive volunteers who are willing to help others as escorts and drop them safely in the time of need. Service requesters can as well act at volunteers. With SafeDrop on their mobile devices, people can view their real-time locations, search for active volunteers around them, send requests through our web service, track the volunteer's location and send instant messages with the volunteer.

We care about users' safety. To ensure the quality of service, SafeDrop maintains a user credit records system and encourages users to rate each other and leave comments after the drop service is finished. In case of emergency, users can invoke fast-dialing using the contact number set up by themselves.

1.2 Target Users and User Benefit

SafeDrop is designed in consideration of anyone especially females traveling alone at night, who are stranded due to unavailability of the public transport or doesn't feel secure to use them at that hour. We assume users have a GPS enabled smart phone with a data plan for connectivity and they are familiar with the basic use of phone and apps.

- People requesting a drop would be able to get to the safe place, and prevent unfavorable incidents.
- People willing to help and be a necessary agent of change would be able to step up and follow a system process to help others.
- The person who is waiting gets a sense of where exactly the help is and how long the volunteer is going to take to reach the place.
- Volunteers receive ratings and can later receive monetary compensation from the sale proceeds of this application if this strategic business model finds funds for initial jumpstart.

2. Features

2.1 Location Tracking

SafeDrop offers user location tracking based on GPS. User opens the application. It opens “SafeDrop Request” screen. The screen shows a map of the current location, with street address and zip code. Location services includes tracking of both the user himself and the volunteer they accept that is coming towards him. There are three main features in location tracking, described as below:

2.1.1 Region of service

To ensure effectiveness of volunteer searching and user matching, we define location into different regions shown in the map by edges. When user cross the edge and enter anther region, those volunteers in that region will be given higher priority to serve.

2.1.2 Shortest Path

Once the requester confirms the offer from a volunteer, a SafeDrop tracking starts. The requester and the volunteer can both see each other’s location on the map and a path drawn to indicate the shortest path between them. This path is designed for auto drivers, since we also expect the volunteer to be safe and have the ability to offer better service.

2.1.3 ETA estimation

As long as the shortest path is confirmed, end users will be given an estimated arriving time of the other one. In case some incidents happen during this waiting period, they both can contact each other and response in time.

2.2 SafeDrop Communication Protocol

For users’ convenience, both requesters and volunteers can make their own choice on their accompany when multiple users are available. Therefore, SafeDrop implements a two-way handshake protocol to initialize an escort transaction.

A complete lifecycle of a SafeDrop escort is :
- Requester sends a request through server to volunteers around him.
- Volunteers that are available receives the request from this requester.
- Requester choose a volunteer and confirm his offer.

- Volunteer that is chosen receives a notification of confirmation. SafeTrack begins.Now users' state swishes to In Process. They can send messages.
- When the requester is escorted home, the transaction will be closed and then archived.

Below is the state model of a transaction:

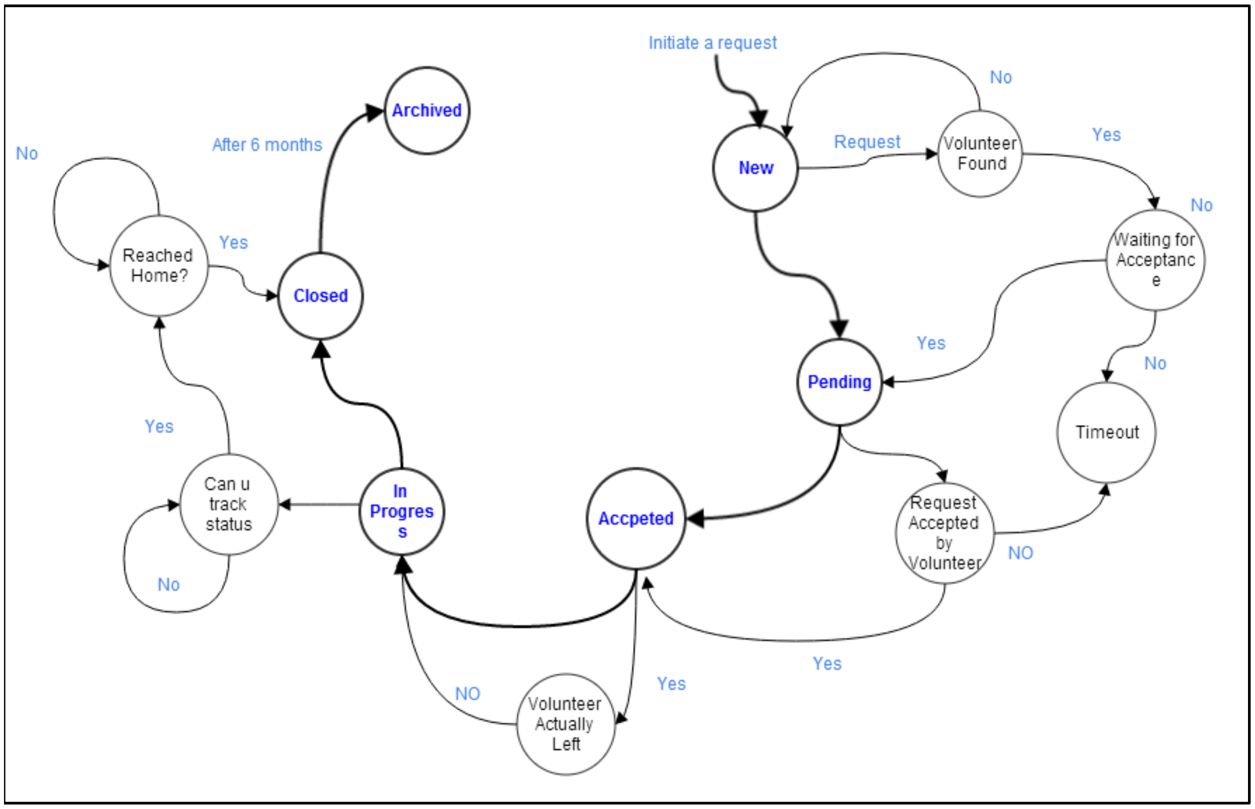


Figure2.0 Request State Chart

2.2.1 Send a Request

User opens the application. It opens SafeDrop Home screen. The screen shows a map of the current location and on clicking on Request button sends a request to the server to enqueue the request.

2.2.2 Pull Notifications

As soon as an request is sent, volunteers within the region will receive notifications. Notifications Icon from where the "Notification" screen is invoked appears with badges and clicks on the “Notifications” icon along with the location map and either a request status or Request button. Screen shows a list of notifications, shows the one which has not been read.

On Acceptance of a SafeDrop Request the requester would receive a notification of Volunteer Accepted, there would be multiple volunteers who would accept, based on which the requester can “View Profile” of the volunteer.

For the volunteer, the status of the SafeDrop is marked Pending for Approval, If marked Accept by the requester, enables the SafeTrack screen.

2.2.3 SafeTrack

SafeTrack shows live status of the volunteer and the requester. It shows a map with pins for volunteer and requester. In this map view, they can see the recommended shortest path and estimate time.
The volunteer/requester can

- Call the requester
- Send instant messages to the other one

2.3 User Credit Management

Each registered user has his user profile achieved in database, which includes a star rating and user comments. Both SafeDrop requester and volunteers can see each other’s user profile and make their decisions accordingly.

Whenever an escort is finished, successfully or not, the requester should rate the volunteer and leave a comment.
The volunteers may get rewards from the sales of SafeDrop according to their rating statistics.

2.4 Peer-to-peer Messaging

We want to provide the best way for the users to communicate with each other. SafeDrop allows the requester and the volunteer to send peer to peer messages through our web service, when users are in the state of “In Process”.

User clicks on Send Message button, it pops out a window represented by message bubble table or bubble list. The messaging task pushes messages to the server and receives messages by pulling from server repeatedly. The sender can also see the status indicating whether the message has been sent successfully or not.

2.5 Navigation Flows

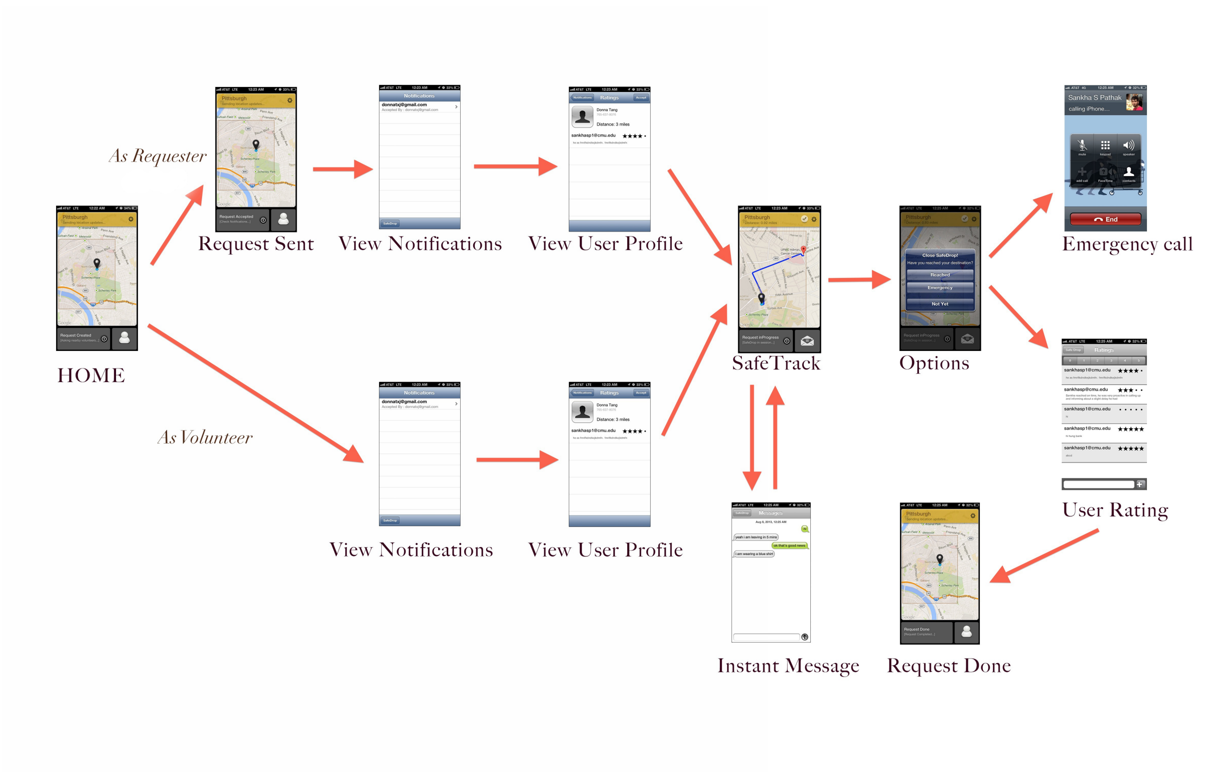


Figure2.1 Navigation Flow

3. Application Implementation

3.1 Challenges and Design Reasoning

3.1.1 Business Challenges

- User registration requires background study: As for the design pattern in current phase, we allow users to register freely. However, considering the specialty of this app, user background research may be required to provide dependable safe drop service. Background research requires reliable user credit statistics from data providers, in addition, we will have to take privacy problems into consideration and enforce strong data security protection policy and tools. Currently we just use pre-defined user data stored in the database, but in the future we may consider integrating with facebook and other large information providers. Furthermore, firewalls and more security tools should be integrated in the system.

- How to guarantee user's safety: Even though SafeDrop can keep track of users, as a requester, he or she may still want to have a backup safety tool in case of emergency. Thus, we provide them with emergency calls. User can set up an emergency contact in SafeDrop settings and once some accident happens, for instance, the volunteer does not drop him or her home, the requester can use one-click emergency dial immediately.

3.1.2 Technical Challenges

Web service architecture:

- As SafeDrop is designed to be an instant helper, we want to enable fast information exchange between users and keep the bandwidth consumption as low as possible. We chose REST&JSON over SOAP&XML, and implemented basic HTTP requests,such as GET, POST,DELETE.

Background task management:

- Location tracking should always be active to ensure smooth workflow. Thus, a worker thread that maintains communication with location service provider and our web server, which updates current user location and push the information to server, should not be interfered by changes of application status. Also, we keep the notification service running once a transaction starts.

- Application state restoration. During the lifetime of the application, any thread may be killed, but we don't want to make this event visible to end users, thus we must implement either state restoration method or other tricks, to ensure whenever user navigate back to the app, the screen shows whatever it was last time. For example, as what we did in iOS application, we tried to maintain the status on a map view. Without stat restoration,when a map overlay is created and we go to another screen and come back, another instance of the NStimer is initiated, which creates another overlay over the map, this makes the overlay darker after a while with repeated rendering.The solution for this is to have an instance of the polygon used for overlay, mark its .map to nil and then do the transition. All map instances should be set to nil and map annotations should be marked nil before doing a transition to another XIB. In the final deliverable, we stored the application state as singleton object GlobalSettings.h in the server to maintain continuous user operations.

User-friendly UI design:

- Reduce UI layers to simplify user operations. As we add more features in SafeDrop, the original UI design became complexed and thus hard to use. For example, if the user's request has been accepted by a volunteer and he wants to view the volunteer's profile, he has to navigate through 4 layers of windows to finally reach the user profile. To reduce the complexity of user operations, we then re-designed the UI architecture.

    - We minimized the buttons in the home screen to provide only available operations according to user's current status. When user's status changes, we re-utilize the buttons and change their responding actions.
    - No more classification page in notification. All types of notifications will be shown in the same screen and user will only see unread notifications.
    - Enhanced SafeTracking Screen. Shortest path and estimated time added to give users more detailed information.
    - Added Instant Message screen and provide fast access on home screen.

- Combining projects using storyboard and xib files for UI. In the beginning, we implemented different parts of the project using storyboard, but soon we came across several merging issues. Then we tried to modify one part into xib files and import the xib files into storyboard, but this is not as easy as it seems as well. Finally, for the connivence of project operation, we changed all UI designs to xib files, and got things work much more smoothly.

3.2 Overall Architecture

This is the end to end solution depicting the below diagram. Whenever a request or any other kinds of data is pushed from mobile end, it goes to internet service provider's gateway, where user can use either the wireless data or local internet.

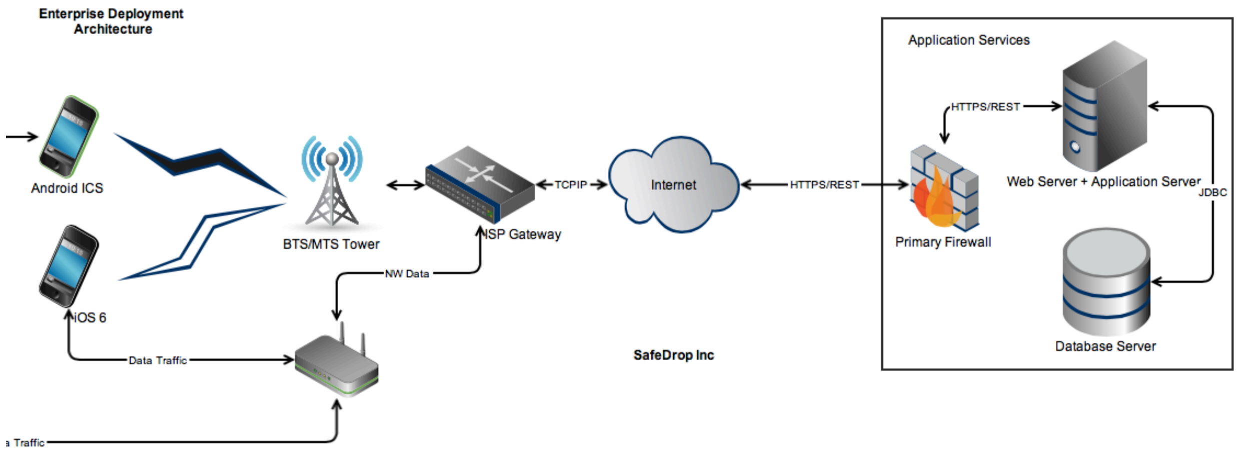


Figure3.1 Overall Architecture

3.2.1 Web services

We use REST services based on HTTP protocol and JSON internet media type exposed by web servers through the internet. REST has been observes with its scalability and lower latency, and JSON is preferred than XML because of its simplicity, readability and light weight.

The business logic resides in the application server. It accesses Database server to get the user details via JDBC connection.We define the data model as described below:

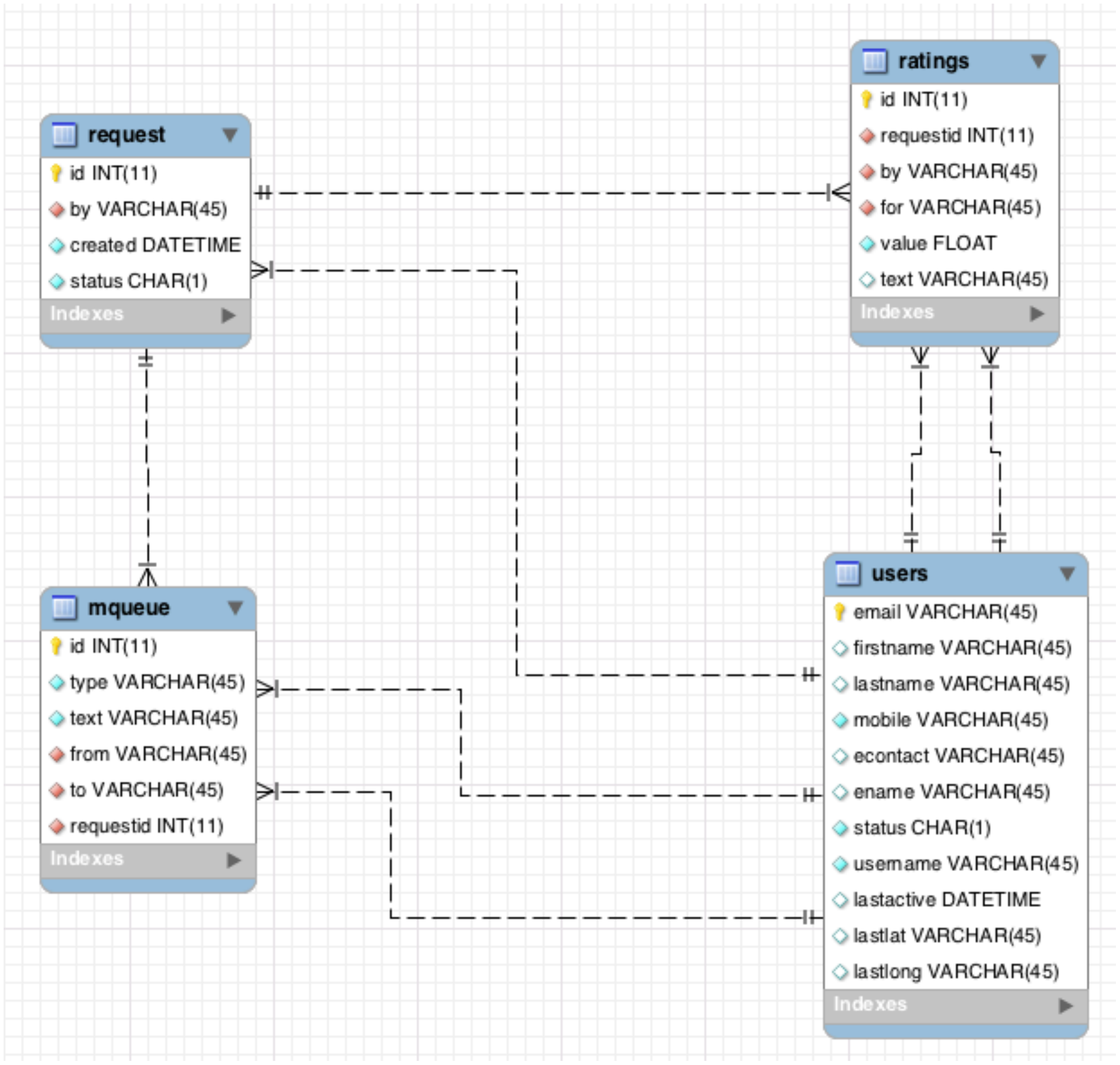
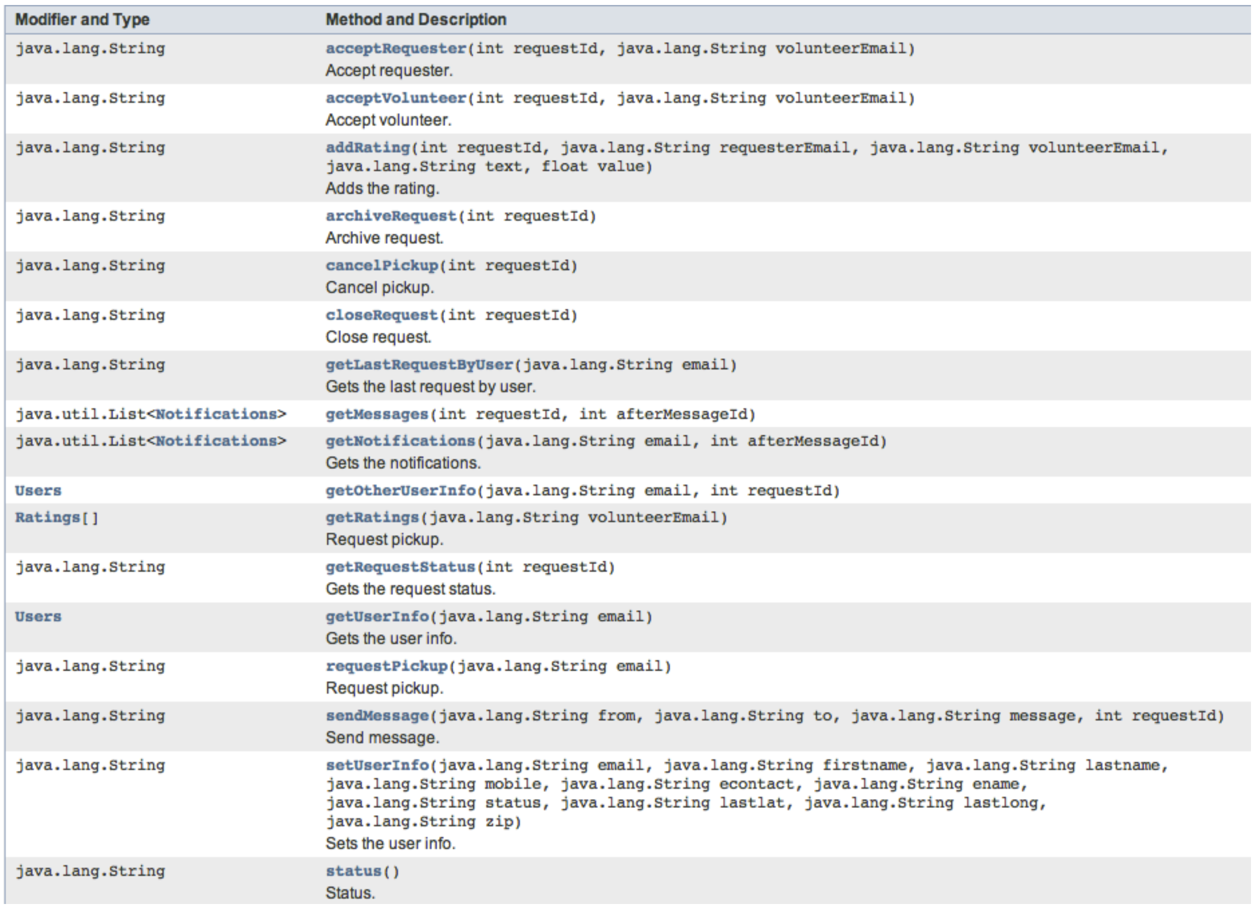


Figure3.2 Data Modeling

The web service we provide includes:



Modifier and Type	Method and Description
`java.lang.String`	`acceptRequester(int requestId, java.lang.String volunteerEmail)` Accept requester.
`java.lang.String`	`acceptVolunteer(int requestId, java.lang.String volunteerEmail)` Accept volunteer.
`java.lang.String`	`addRating(int requestId, java.lang.String requesterEmail, java.lang.String volunteerEmail, java.lang.String text, float value)` Adds the rating.
`java.lang.String`	`archiveRequest(int requestId)` Archive request.
`java.lang.String`	`cancelPickup(int requestId)` Cancel pickup.
`java.lang.String`	`closeRequest(int requestId)` Close request.
`java.lang.String`	`getLastRequestByUser(java.lang.String email)` Gets the last request by user.
`java.util.List<Notifications>`	`getMessages(int requestId, int afterMessageId)`
`java.util.List<Notifications>`	`getNotifications(java.lang.String email, int afterMessageId)` Gets the notifications.
`Users`	`getOtherUserInfo(java.lang.String email, int requestId)`
`Ratings[]`	`getRatings(java.lang.String volunteerEmail)` Request pickup.
`java.lang.String`	`getRequestStatus(int requestId)` Gets the request status.
`Users`	`getUserInfo(java.lang.String email)` Gets the user info.
`java.lang.String`	`requestPickup(java.lang.String email)` Request pickup.
`java.lang.String`	`sendMessage(java.lang.String from, java.lang.String to, java.lang.String message, int requestId)` Send message.
`java.lang.String`	`setUserInfo(java.lang.String email, java.lang.String firstname, java.lang.String lastname, java.lang.String mobile, java.lang.String econtact, java.lang.String ename, java.lang.String status, java.lang.String lastlat, java.lang.String lastlong, java.lang.String zip)` Sets the user info.
`java.lang.String`	`status()` Status.

Figure3.3 Web service chart

3.3 iOS Development

3.3.1 Architecture

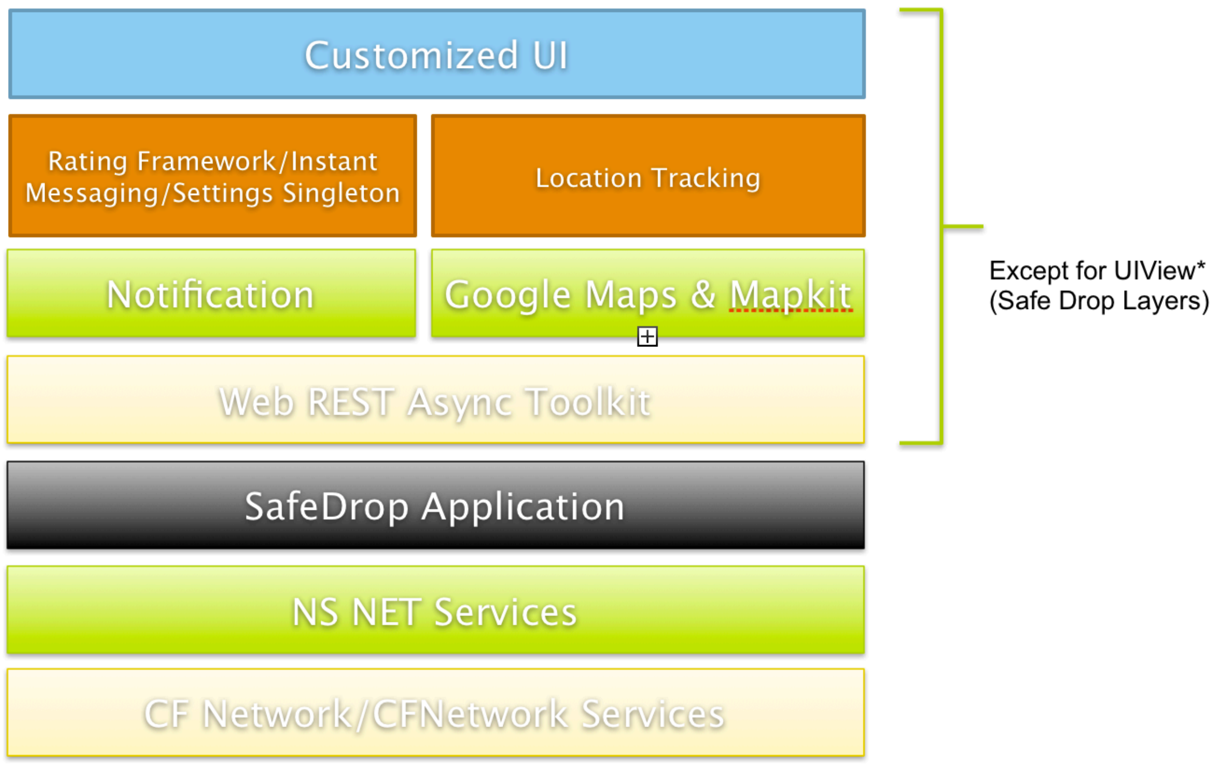

Customized UI
Rating Framework/Instant Messaging/Settings Singleton
Location Tracking
Notification
Google Maps & Mapkit
Web REST Async Toolkit
SafeDrop Application
NS NET Services
CF Network/CFNetwork Services
Except for UIView* (Safe Drop Layers)

Figure3.4 iOS App Architecture

Base on iOS core services, primarily the CFNetwork framework and Location framework, we built up the SafeDrop application.

The REST API built with Cocoa class NSURL that relies on CFNetwork, and since iOS5 and the above version have built-in JSON parser classed, thanks to which JSON media can be extracted and parsed easily by using NSJSONSerialization class and contains the data in different types of NS JSON object classes.

The location service provides the features described above, implemented mainly with GoogleMap framework. We use Google Map view to show locations, and the reverse-geocoding is done by CLGeocoder. Zip code region finding and shortest path drawing are implemented with google distance API.

To build up notification and instant message service, we use Grand Central Dispatch method to manage background processing and invoke the main UI thread again to update UI views.

3.3.2 Tools used

- GoogleMaps & Google Distance API. Used for MapView displaying, locationing, shortest path search, zip code region definition, etc.

- OpenGL ES. Not included in the final deliverable, but used before to support 3d buildings in high zoom level.

- QuartzCore. Used for UI effect rendering. Most important UI effects such as drop shadows, rounded edges, and transparent views are based on QuartzCore framework. It is low-leveled but quiet flexible and easy to use.

- TPAvoidScrollView. Used for keyboard animations in messaging window. It allows the keyboard to push up the current view on the screen and enable users to type in messages without visual interference. After typing, the keyboard will slide down to the bottom.

- AMRatingControl. Used for user rating and review section. Provides decent UI and rating controls.

- UIBubbleTableView. Used for the layout of instant message window.

3.3.3 Lessons Learned

- Using Google SDK v2 for iOS
- Integrating REST API and use web service.
- Storyboard and Xib together.
- Background task managing by Grand Central Dispatch
- Application Status Restoration methods.
- Pushed Notifications (Not implemented in the final deliverable)

3.4 Android Application Architecture

3.4.1 Architecture

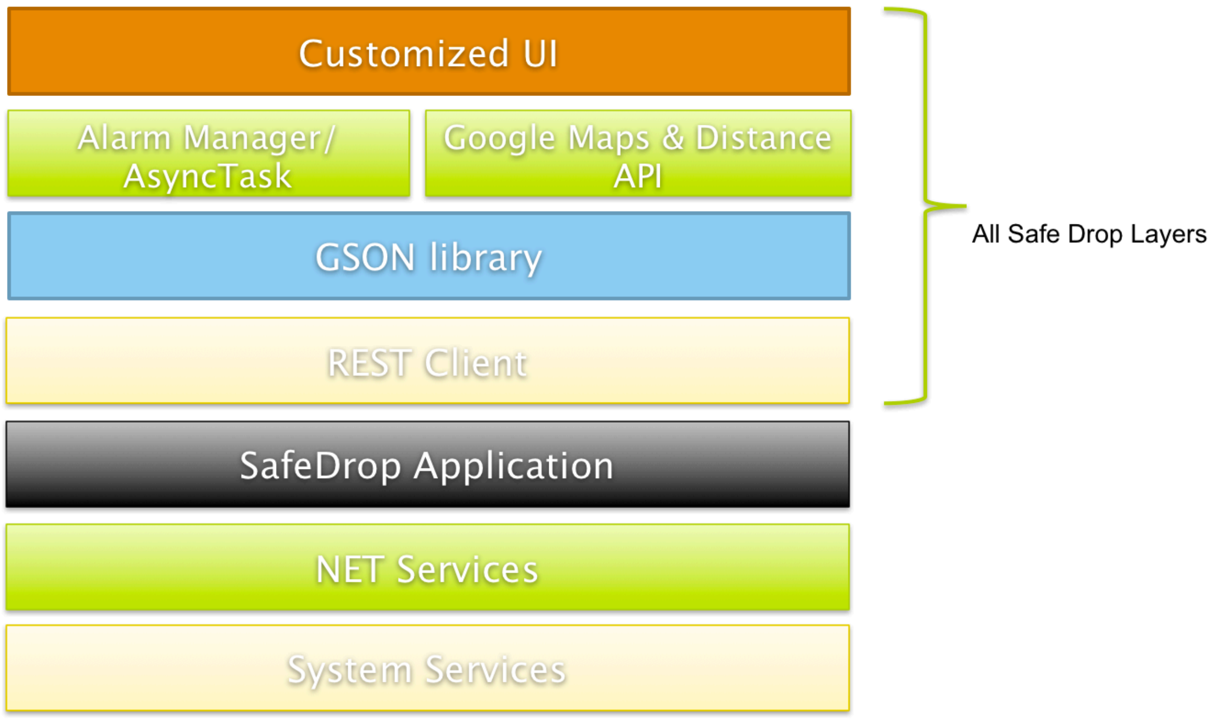


Figure3.5 Android App Architecture

Compared with iOS app architecture, the Android version utilized the GSON library for JSON parsing in communication with the web server.

As for the customized messaging window, we designed our own bubble view with separate layout files and customized bubble icon with android ADT 9-patch drawer.

For notification and message exchange, we also implement background tasks by using AlarmManager instead of timer tasks, since we require the notification thread always running as long as the phone is powered on. While timer tasks are light-weighted, it will stop when the phone goes to sleep mode.

3.4.2 Tools Used

- GoogleMaps & Google Distance API. Same as in iOS, these are used for MapView displaying, locationing, shortest path search, zip code region definition.

- Google GSON. Used for JSON parsing.

- Bubble View & 9-patch drawer. Used for building the UI of instant message window. To use out own designed bubble icons, we used 9-patch drawer, an Android ADT tool, to generate bubble backgrounds that are scalable according to the length of message data.

3.4.3 Lessons Learned

- Using Google SDK v2 for Android.
- Integrating REST web service, wrap APIs by AsyncTasks to deal with HTTP connections.
- Customization of ListView to show message bubbles. Designed separate layouts for list view cells and bubble background images.
- Using AlartManager instead of Timer and Timer Tasks, to ensure continuous background thread execution.
- Applying different activity launch mode accordingly. Managing the resources and threads based on activity life-cycle.

3.5 Comparison between iOS and Android development

Apple always wants to provide users and developers an easiest and meanwhile specified and standardized way of getting things done.

The iOS SDK is a governed system that is centralized on consistency, with several built-in frameworks, the developers' workload is often minimized. For example, core location service, SQLite database service, Core Networking service are all provided. All these frameworks can coordinates with each other smoothly and provide developers with simple and quick API.

Xcode, on the other hand, is quite easier for developers especially starters to use. For example, mapping UI controls with actions or outlets can be done by just clicking and dragging. Relative code will be added automatically for users. The iOS device emulator works faster and the debugging procedure is simple. Also, User can adjust IDE windows with one-click on the panel. The editing view automatically opens the header file for the current file. Those features all make Xcode an ease.

Google, as it always does, provides developers a free space of create, design and implement their ideas by different solutions available. In short, Android development is OPEN, which enables developers to do thing in their own way.

Android SDK uses Java-based language, which easily attracts a large number of developers. In addition, based on large market proportion, Android developers can take advantage of mass amount of documentation and references on line. Google also maintains the docs pretty well. Because it is open sources, it updates very fast and it's super hard to maintain continuousness between different versions. Thus the fragmentations may be significantly large and contributes to the difficulty in developing applications that are targeting various API levels and devices.

Eclipse IDE may be easy and the best choice for Java programmer to use, however, it may be difficult for starters to use.

4. Future Work

Due to the limitation of time, there are still work to be done both for iOS and Android application.

- Supporting of more models for iOS and supporting of various target levels and screen size and resolution for Android. More efforts on combining code for resolving fragmentations.
- Push Notification. This is the most effective way of sending and receiving notifications. Currently we are maintaining a running thread to pull data from server, however it is consuming lot of resources of the device. With push notifications enabled at server end, the notification center on the device can be utilized thus avoiding too much resource consumption and making the project simplified and robust.
- Integration with Facebook as an option for user background study and social network features to improve connectivity among users.