**State University of New York at Buffalo**

**Department of Computer Science and Engineering**

**CSE 646 - Wireless Networking and Mobile Computing**

Project Report

***Friend in Need***

Project Blog - <http://cse646.blogspot.com/>

Developed by-

**Amit Nihalani** (50133954) (amitpuru@buffalo.edu)

**Rohit Dubey** (50134121) (rohitdub@buffalo.edu)

**Vivin Rane** (50134206) (vivinran@buffalo.edu)

1. **Abstract** -

With the advent and popularity of smartphone technology, they’re finding new applications in almost every aspect of our life. It’s easier than ever to stay in touch with friends and family in this connected global village of 21st century. However, so far the innovation has been more focused on enhancing the ways to connect with people who we already know. Often times, we need help nearby, and in some of these cases, people other than our friends are better placed to help. Examples range from helps during emergencies to help regarding specific aspects of our life from relatively expert people. We lack a medium that allows trading helps between people. As a result, sometimes, the help we’re looking for is steps from us and we still keep reaching out to people who might be far away. We present the app, “Friend in Need” designed specifically for the purpose to resolve this. The app allows a user to post the kind of help they’re looking for within their geographic vicinity and available users to respond to them, thus attempting to create a connection between people seeking help and people who can help.

1. **Introduction and rationale**

Our android app seeks to serve as a connection between people seeking moderately to extremely urgent help and people available in a position to help them. Possible scenarios range from emergencies to help with day-to-day activities. a few examples are: car breakdowns/Accidents, bad weather shelter, natural disasters, contacting emergency services in the event of an accident, loaning low cost items such as books and phone chargers, providing opinions/ideas on projects, help with setting up electronics etc. “Friend in need” attempts to keep the nature of helps as general as possible to avoid become yet another specialized resource in a specific area (study group, real estate, help forums etc.). By supporting a far wider spectrum, it improves the chances of availability of help exponentially since a user able to offer help in setting up electronics might also be able to loan a phone charger to someone with exhausted battery.

1. **App Design**

Our main focus of developing such a collaborative app was that app users wanting urgent or immediate help are not left in despair and can contact enthusiastic volunteers in their vicinity who are ready to help. Therefore the interface needs to make it extremely and quick to post help. Due to this, most of the information about the help is automatically added to it (User, time created, Location etc.). All an “initiator” has to do is to type in at most 140-char message describing the nature of help and click the post button. The app takes care of including additional data, uploading it (even in intermittent connectivity) and making sure other users in the vicinity are able to see the help as soon as possible. To provide these functionalities, we use components such as Facebook authentication and Google Maps API to include these aspects of user information with a post.

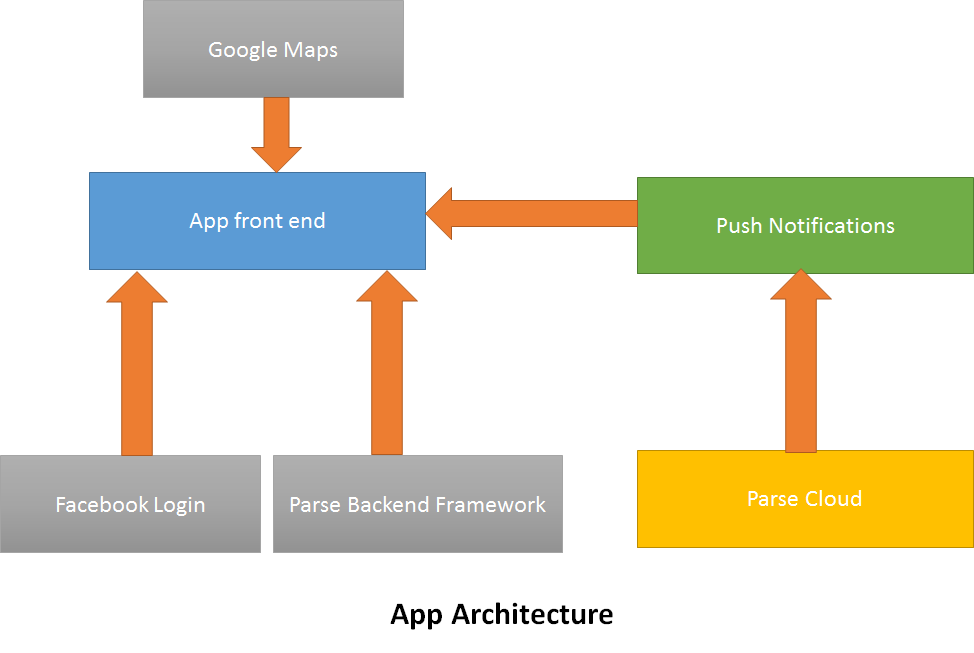
Similar optimizations exist on responders end as well. We provide List as well as Map view of the nearby helps with ability to change search radius depending on a user’s ability to travel. The users are also shown helps posted right outside their search radius (with only the location information) since there’s not always a hard boundary to a region that a user can provide help in.

A responder can respond on a help post with a short message detailing his offer to help the initiator and preferred mode of further communication (text, email etc.). We view this functionality as a medium for an initiator to receive initial offers from volunteers and then taking the conversation to a preferred mode of communication rather than restricting them to an inbuilt chat/messaging interface.

The user authentication is quite important in such an app and we chose to do this solely using Facebook since it provides much more detail about a person than any other social media platform and we certainly didn’t want users to fill up a lengthy form during signup and verify their email ids, phone numbers etc.

1. **App Architecture –**

We chose Parse API as our backend interface since it fits so well in the android framework and encapsulates much of the details required yet providing powerful functionalities. The front-end of the app is developed on Android studio, while the backend is handled by Parse framework including data and functions for push notifications (JavaScript).In addition, we use Google Maps API is used to view a user’s location and find nearby users in required radius. Facebook login is used for authentication eliminating the requirement of additional usernames and passwords.



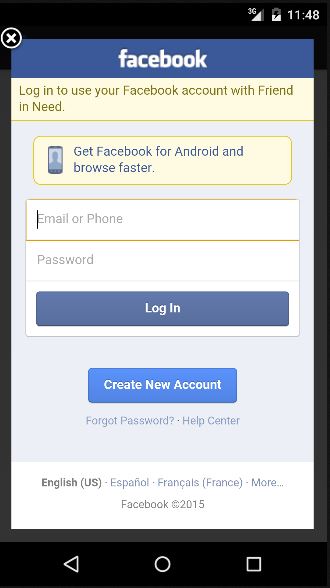
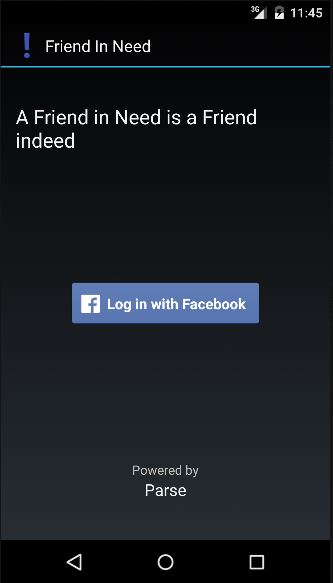
* 1. **Parse Backend**

Parse API, owned by Facebook, is a comprehensive framework for managing Cloud core, push notifications etc. almost on any platform available (Android, iOS, Windows, Unity, embedded etc). It’s one of the fastest growing cloud solutions available and provides native SDKs with powerful functionalities built on top of native classes to abstract away details. For example, saving data to the cloud using Parse is just a matter of saving (key, value) pairs in a *ParseObject* class instance and calling powerful methods like, *saveInBackground, saveEventually* etc. to save them to the cloud even in intermittent connectivity.

We use 3 databases in our backend namely, *Messages*, *User* and *Posts* in addition to automatically generated *Installations*, which records details about every installation. In addition to the data, we also chose to implement certain functionalities like Push Notifications in the cloud to security risks in sending Push notifications directly from one app instance to another. Parse push notifications uses Google Cloud Messaging (GCM) to send push notifications on Android and the cloud function is called in the background to authenticate the sender, verify the message and send out push notification to the recipient.

* 1. **App Components**
     1. Facebook Authentication (WelcomeActivity)

We use Facebook integration as a sole medium for users to log in to the app due to concerns outlined before. Our app launches the Facebook android app in the device or a browser session in case the app in not installed on the device. Once signed in, the user details are cached to avoid manually logging in the next time the user launches the app. We only request basic data such as user name, email and public profile from Facebook and don’t post anything on the user’s wall.



*FIG: The welcome screen and facebook log-in*

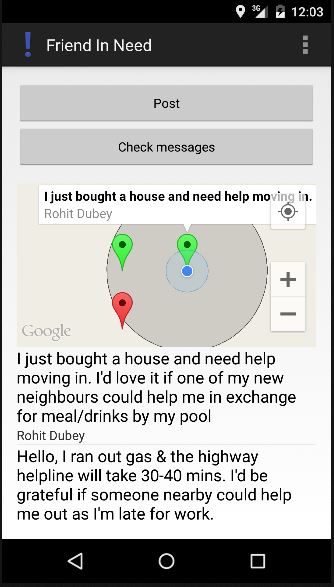
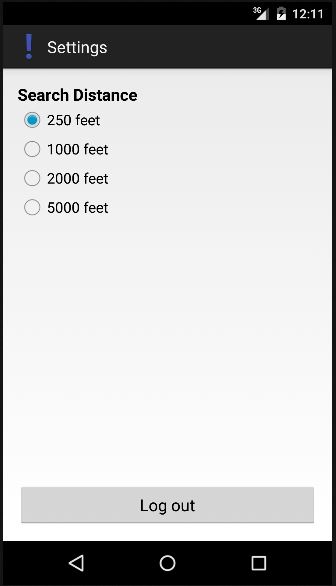
* + 1. HomePage (MainActivity)

Accessing and using a User’s location is fundamental to the purpose of our app. We use the Google Maps API to provide basic location related functionality. The Map fragment on the homepage provides a view of Posted helps in the area along with small *infoWindow*s to send a message to initiator. We also present the posted helps in a list, sorted by most recent, to make it easier to view the helps, even in a slow internet connection.

The posted helps just outside a user’s search radius are also shown using red markers (without any description) since the search radius of users isn’t always a strict limit and most users are fine helping out people a few feet outside that radius.

A user can always increase their search radius the Settings if they can cover a larger area. The settings page also provides a Logout functionality. We refresh the Map results every time the activity is launched and when the map perspective is significantly changed by dragging along the Map.

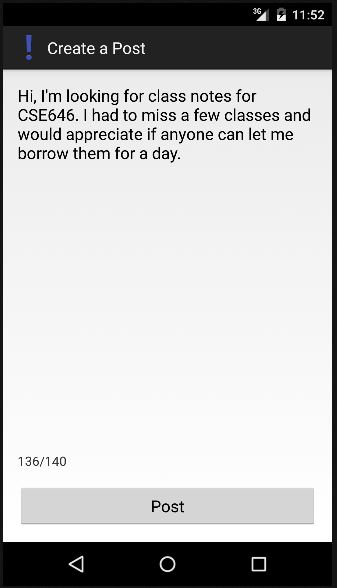
The infowindows on the map can be clicked to allow a user to contact the initiator.



*FIG: APP HOME AND SETTINGS*

* + 1. Posting new Help (PostActivity)

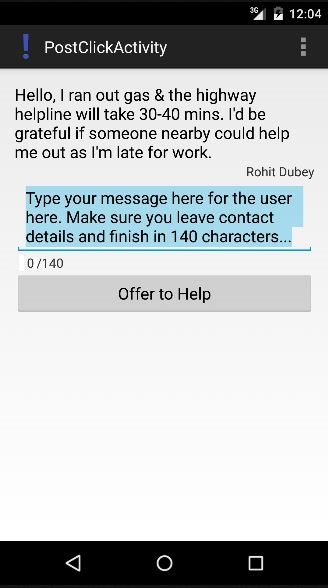
A user can post new help by clicking the Post button on the homepage. Posting only needs an at most 140-character description about the nature of the help. All other relevant information such as, sender’s name and location and time of posting is recorded automatically.



*FIG: POSTING A NEW HELP*

* + 1. Volunteering to help

A user can volunteer to help on any of the posted helps by clicking on the help post on the app homepage. This launches a page, where the volunteer can describe the help they can provide and preferred contact method for further communication.

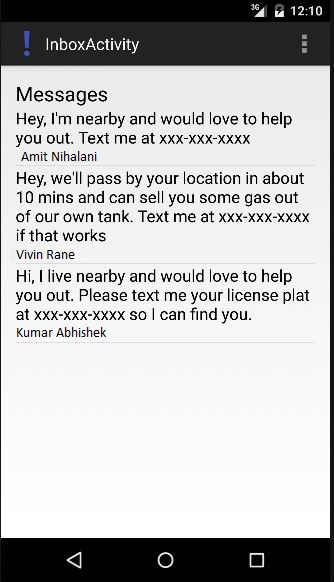


*FIG: OFFERING TO HELP*

* + 1. Reviewing help offers

An initiator can consider the help offers by reading their messages. The messages view is accessed by click the Messages button in the homepage.

The messages view brings up a list of messages by the people who’ve offered to help sorted by the most recent first along with their names. An initiator can then review these offers and contact one of the volunteers using the preferred contact details provided in the message.



*FIG: HELP OFFERS*

1. **Smartphone Usage -**

The app will use smartphone as medium of locating nearby users. We access GPS location of the user to connect the initiators and helpers. The smartphone provides a medium of convenience and along with Push notifications increases app usage, hence increasing the chances and quantity of help offers. The app functionality and convenience would have been extremely limited if it was implemented on a Desktop or Web platform. The app also needs a working internet connection to work but is optimized to work in slow or intermittent connectivity. The techniques to cache a user’s data and save objects on the device till an internet connection to upload them to the cloud is available allow the app to be used in a variety of scenarios. Also the app can function even if the GPS location of the user is unavailable. In this case the app uses the relatively less precise location available without the GPS. This feature, combined with minimal internet access can help save battery in emergency situations.

1. **Challenges Faced-**

In every phase of the app development, we had a few issues that took significant effort to solve. Following points detail the major issues we faced and ways we solved them.

* 1. **Coordination between Parse and Android**

One of the most challenging aspects of the final development phase was the integration of all the modules. Since we are using Parse for handling the backend and Facebook for authentication, our app had to be consistent with the class objects that are used in Parse, Android as well as Facebook (wherever applicable). For example, while sending user notifications, we have to map an android user to their Parse id and user the Parse user ids to identify the users and send them notifications. Parse users also have to be mapped to corresponding Facebook users Parse made this task relatively simple since we were able to follow many Parse tutorials and build upon them. Parse’s popularity certainly helped us to find help on these aspects online.

* 1. **Transition from homepage to message page**

We needed a way for users to be able to message an initiator easily without breaking the user experience flow. We wanted both the list and the map on homepage to allow a user to transition to the message sending page. This involved making the infoWindow on the MapFragment object clickable. However, we could only find this attribute available for Map object, which would need our app homepage to show a full Map instead of MapFragment and a ListView. Fortunately we were able to obtain the Map object from the MapFragment using *getmap()* method and use Map object methods on it. The only alternative would’ve been to customize the entire infowindow, thus increasing app complexity considerably.

* 1. **Facebook login issues**

Due to an existing bug, the Parse class ParseFacebookUtils returned a null user when signing in using Facebook, especially on devices without the Facebook app installed. Even the devices with Facebook app had missing user attributes. After much searching, we had to resort to using the previous version of Parse API instead of the latest. This in turn, forced us to use previous version of Facebook android SDK due to some methods being deprecated. As a result, we had to make changes to the entire app code to make up for the methods not available in the previous version. A drawback of this approach was that the ParseUser returned after logging in through Facebook contained a Parse generated string as the username instead of user’s name on Facebook. We had to execute an extra GraphQuery to obtain the user’s name and save it as the current user’s username before they could Post any helps or reply to any of the initiators.

1. **Timeline**

We used integration thread approach to develop the app, wherein we first implemented basic features to make a working app and successively added features. We had planned to complete UI design and basic app framework in the first month of development. Next was social media integration app. The remaining time was spent on testing and debugging and adding additional features based on the feedback during Initial and Interim presentations in class, such as Messages, Push notifications, Search radius customization etc.

1. **Future Prospects**

With the first version of app fully functional, we’re looking to implement new functionalities along the following directions to enhance the user experience and help Friend In Need better serve its users

* 1. **Urgency levels of helps posted**

Some of the helps posted might be more urgent and time-dependent than others. We plan to capture this aspect to make sure initiators requiring urgent helps are shown with priority to other users. However, to implement this, we need to design a way to avoid malicious use of this feature to tag non-urgent helps as urgent to get priority treatment.

* 1. **Posting on Facebook**

It might be helpful if an urgent help post could be, with permission, shared on user’s Facebook profile to reach more audience depending on urgency levels. This could also increase app popularity. However, to implement this, we need to implement 8.1 above to make sure, not every help is posted on a user’s Facebook and review of our app by Facebook team since we’ll be using more permissions than the basic permissions.

* 1. **Sharing pictures**

It would enhance user experience and help them make new friends if pictures of the experience could be shared on a user’s social media accounts. This could work by app sensing the proximity between and an initiator and helper and prompting users to remember to take a few pictures and share it with their friends. This feature, while not in the first development cycle, since it required considerable effort which was not feasible in the timeframe, could certainly be incorporated in future releases.

* 1. Integration with Google Play Games

To provide an added incentive for users to help each other, the app could be gamified and be integrated with Google Play Games API to generate leaderboards depending on number of helps to increase competition and hence helps provided. However, this would need us to design a way to make sure the system is not spoofed by fake helps and a way to confirm a helper indeed helped an initiator along with a way to evaluate helps.

1. **Individual Contributions**

Since all the team members were relatively new to Android, we followed the approach of everyone studying the basics of every aspect and then assigning the 2 most confident team members per major task to encourage knowledge sharing and better brainstorming. The third person was kept up to speed on progress and helped occasionally in case of issues. Everyone took turns in posting the updates on the blog. This way, all three of us had some contribution in every component of the app.

|  |  |
| --- | --- |
| Parse Backend and Android core | Rohit and Amit |
| Cloud code (Push Notifications) | Amit and Vivin |
| Facebook and Google Maps Integration | Rohit and Vivin |
| Documentation, Presentations and Blog management | Rohit, Amit and Vivin |