FriendGuard: A Context-Awareness Mobile App Helps Securing College Female Safety

Xiaomeng Jiang
University of Maryland College Park
jackiezjiang@gmail.com

***Abstract*—The prevalence of sexual harassment, sexual assaults and rapes has brought social attention across the world. US government, universities, companies and groups all make significant efforts preventing such events [1].To help reduce sexual affends and crimes, and help securing college female safety, we study the problems of existing safety apps, design and implement a new Android app FriendGuard. It is a context-awareness app that enables location sharing and status updates through shared session with friends. By sending notifications through alerts or check-ins, user's status will be shared to the friends, who can take proper action to help and prevent possible sexual crime when necessary.**

***Keywords*—Context-Awareness, Safety App**

I. Introduction

Female students are on the rise of sexual harassment risks, even though a growing body of efforts has been make to prevent it. Campus Law Enforcement has been issued by U.S. Department of Justice[2], to prevent violence and crime as a whole, and to reduce sexual crime as well. Many Universities and Colleges in US also publish anti-harassment policies as a way to reduce sexual harassment or sexual assaults[3].

However, according to Association of American University (AUU) 's survery, the incidence of sexual assault and sexual misconduct happens to 23.1 percent of surveyed students across 27 universities throughout US [4]. Other survey shows that among all the rape or sexual assault experienced by females between 1995 to 2013, 33% were completed rape, 25% attempted rape, 31% sexual assaults, and 11% threats [4], [5]. Behind those high percentage, one also has to notice that the response rate of surveys are between 20% to 30% [6], [7]. That is, most of the females, victims or not, are reluctant to report sexual assaults or harassment [4]. On one hand, it means that even though we have many anti-harassment policies and offices available, they may not serve well when unpleasant things occurred. One possible reason is that, females don't want to report and share their feeling with someone they are not familiar with, and they would rather keep it secret. On the other hand, that means the percentage of female victims may be under-estimated.

What also worth noticing is that, most sexual crime occurs when victims were pursuing leisure activities away from home [5]. Therefore, we should not only enforce safety policies and actively prevent sexual harassment/assaults on-campus or at working places, but also need to actively educate students and reduce the off-campus occurrence as well.

To reduce the on-campus and off-campus sexual harassment and assaults as a whole, we need to answer the following questions.

- How to increase the report rate? It will not only provide us more detail insights and statistics, but also will help females speak out when something unsafe or unpleasant happens.
- How to protect privacy? The more privacy we can ensure, the higher reporting rate we can expect, and the more likely it will help preventing sexual harassment and assaults.
- How to achieve timely report? If we can make report close to real-time, then many of the crimes can be prevented, or stopped before it gets worse.

We here present FriendGuard, an mobile app that allows location and safety status sharing with family and friends. It aims to increase the sexual harassment/assualts report rate, protect user privacy, by letting users connect and notify their close friends or families only. It also enables real-time report through a shared session between user and their friends. Note that, FriendGuard is designed to help preventing sexual harassment and assaults, but it is by no means can serve as an emergency app, or a substitute of 911. That is, when emergency happens, one should always call 9-1-1 for help, rather than relying on our app.

In the following sections, we use notation User and Friend, referring users to be FriendGuardED and to be FriendGuardING, respectively.

II. Related Works

There are existing applications targeting female safety [8], among which Companion, bSafe, Drunk Mode are popular among users. We therefore review their features and compare our work with those existing apps in the following section.

According to Companion description [9], it is a "personal security solution" with the help from connected friends and families. Users can share location with chosen friends or families, and fire alarms to them when emergencies occur. It has 2 alert types based on severity of events. One can send "I am nervous" or "call 911" directly. Companion also asks user to confirm their status at certain time interval. This will help firing alarms when they dropped the phone.

bSafe is another very popular safety app with many media exposure [10]. It has rich features: notifying friends, fake call,

and share location, et al. Customer has many options when something unpleasant occurs. And the more a customer gets familiar with the app, the better it can serve the customer. However, it may involve too many human interactions, which will delay the reporting time and make operations not simple enough for an emergency response.

Drunk Mode [11] focuses more on party safety, by tracking drunk user. It can prevent drunk if necessary, and will also help a friend to locate and reach a drunk user after a party. Even though it seems a party-specific app, it can also serve as a safety app in general.

All the above apps have several common features: record user location and sharing status with friends. FriendGuard bears those key ideas, and explore further. We makes our app more proactive by involving user safety confirmation at each active check-in time point. We define our target users as college students. The narrowed target of FriendGuard brings a favorable feature - the college friends may know the location of FriendGuard user much better than a remote friend. The background information from friend will increase the possibility that they make a best decision to help, and come to the shared location in the shortest possible time. Also, we provide only two simple mode: Alarm or Check-in, and keep the user input to a minimum. This will on one hand make our app simple and easy to use; and on the other hand, let user send alarms as quickly as possible.

III. Design and Implementations

A. Context Awareness

In Context Awareness systems, Context is defined as ”Any information that can be used to characterize the situation of an entity” [12]. FriendGuard enables Context Awareness in a slightly different way than most of the Context Awareness systems. Our definition of Context is ”the combination of location, time and safety status, along with human decisions”. For User, context will determine when and what to be sent to Friend. For Friend, context will provide information so they can decide how to help User. The involvement of human thinking will allow best flexibility and most proper response to User from their friends.

B. External Dependencies

The way our application provides protections for female college students is through (1) sharing locations with friends and family using (2) notifications. Therefore, the following dependencies are used in our app:

- Google Map API: This is how we share locations between user and friend, and how we keep track of the location change of users. For both User and Friend, a map is displayed through Google Map, so the shared location can be visualized.
- Google Messages: It will relay the message between user and friend with minimum delay. Therefore, all the notifications sent are close to real-time.

C. Features

1) Create A Session: A Session is a connection within certain time range, where the Users share their locations and safety status with their Friends. It can be created anytime by the Users by specifying **Start Time**, **End Time**, **Check-in Intervals** and a list of friends from **Contacts**. Then all the friends invited will receive invitations about the very session. Once the Friend accepts invitation, a connection is successfully built, so now User’s location and safety status can be shared with Friend.

2) Sending Alerts: Alerts are sent manually by Users when they feel unsafe or in a situation where they feel uncomfortable with. By simply pressing the big alert button in the app, the User will be able to send a “In Danger” notification along with their exact location to the Friends. Once the Friends receive the notification, they can open and view the User’s location (when Alert is sent) pinned on the shared Google Map. Then, the Friend can respond by contacting User, coming to User’s location to help, or directly calling the police if necessary.

3) Active Check-ins: As compared to Alerts, Active Check-ins will proactively check the safety status of User by prompting notifications at the end of each time interval. The User can confirm their safety. Or if they missed the check-in after 5 min, a “In Danger” notification will be sent to the Friend. After that, if User realizes he/she missed a check-in, they can notify Friends by setting the status back to ”Safety”. Being safe or in danger, the exact location of User will be sent and stored for later retrieval.

D. System Design

The FriendGuard system is partitioned into Client side and Server side. Server side includes the back-end data storage, while client side is the actual application. Based on the role of FriendGuard app users, user is categorized as either User (who would like to be FriendGuard**ED**) or Friend (who does FriendGuard**ING**). Both User and Friend have to be registered to be able to interact with Server. And Friends will be restricted to those on User’s contact list. This will ensure the privacy of User’s status, and also increase their willingness to report the unpleasant sexual harassment or assaults.

We will explain in details about interactions between Client and Service for our key features.

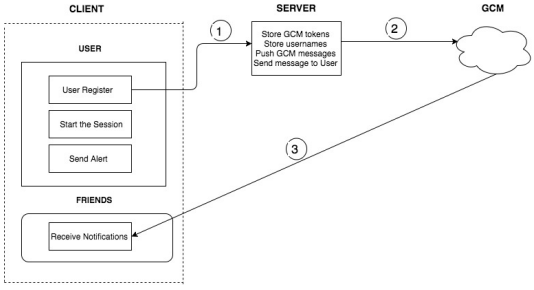

Fig. 1: Data flow of User Registration

1) User Register:

1 When a User registers with FriendGuard, the Server will request and get a Google Cloud Messenger (GCM) token by interacting with GCM service. At the same time, User's information and the contact lists from their mobile device is also uploaded to the Server.

2 The Server stores information of User to the local database, and find all the registered Friends' information. Once the registration succeeds, it will send notifications to GCM, asking broadcasting of the registration success message.

3 GCM will relay the message sent from Server, so User and Friend with receive registration notifications. Behind the scene, User and Friend will receive GCM token to be used for messaging in their session, so they have the token to talk to each other through GCM.

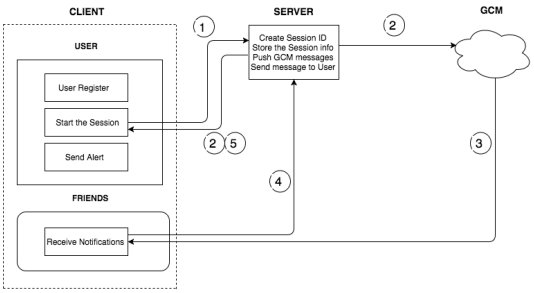

Fig. 2: Data flow of Start Session

2) Create A Session:

1 To create a Session, User needs to specify a start time, end time, friends to be connected, and check-in interval through the app. All those information will be sent to Server, and will be stored in the database.

2 The Server will create and assign a unique sessionID for the new session. The sessionID is then sent to GCM, so it can be distributed to all Friends.

3 GCM sent invitation notification to Friend for the newly created session bind to sessionID received from Server.

4 Once a Friend accepts the invitation, the sessonID will be shared between User and Friend. Therefore, a connection is build. Then the confirmation is sent back to the Server and get stored.

5 The Server sends the agreement to the User via GCM. And User will receive notification for invitation accepted and confirmed, and he/she gets the FriendGuard ready to work.

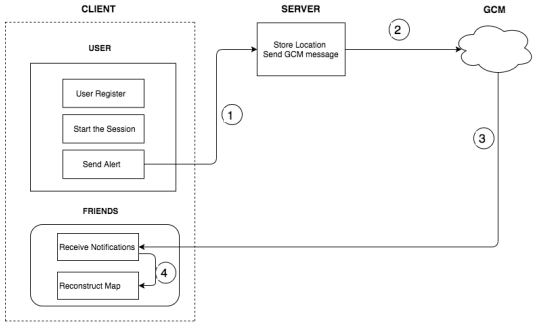

Fig. 3: Data flow of Send Alert

3) Sending Alert:

1 Sending alert will be triggered in 2 cases: (1) When User sends an safety alert by hitting the alert button in app (2) User misses a response to Active check-in for more than 5 min. In both cases, the time, location, and sessionID will be sent to the Server.

2 Server will store the information sent from User, and sends a message to the connected Friends via GCM.

3 Friend receives a notification for the User's alert.

4 Opening the alert notification, Friend is able to see User's exact location when alert is sent, along with the time and safety status in Google Map.

E. UI Design

The main purpose of our app is to let the User respond to unsafe situations as quickly as possible. Therefore, our design focuses on

- minimizing User input once the session starts
- making the user input as easy as possible.

a) Start A Sessioin: To simplify the process of sessioncreation, we use scrolling menus for check-in interval input,and Google Datepickers for start time and end time input. Theinterface for session creation is shown in Figure 4

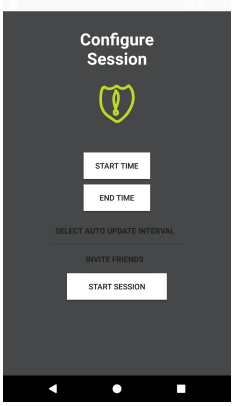

Fig. 4: Start a new Session

b) Session View: Fig 5 shows the interface when User is in a active session. It has a big ALERT button, so the it can be easily found and pressed in unsafe situation by User. Also, the map in the session view will allow the User to visualize their location.

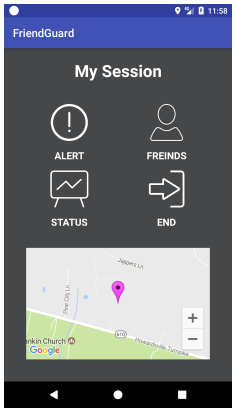

Fig. 5: Session View Interface

c) Friend View: Once alert notification is sent by User, Friend's device will vibrate when the notification arrives. This will help make sure the Friend will not miss the important update from User. If Friend opens the alert notification, he/she will see the message as shown in Fig 6. It states that User is sending ALERT, and the exact location is shown in the map.

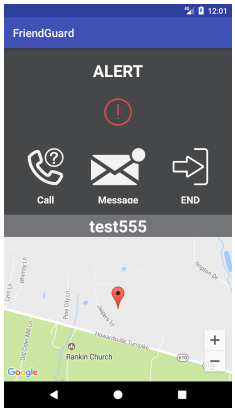

Fig. 6: Friend's view of a User alert.

IV. Conclusions

We present a context-aware mobile app helping prevent sexual harassment and assault of female college students.

It provide features allowing users to share location, safety status to friends. Also, we introduce an active check-in feature, so the users can confirm their safety status at certain time interval. Having this, the user is able to actively check the surrounding environment and make sure they are safe, to periodically share their status by simple click, to be tracked when something happens and they cannot update the status.

Our target users are female students, so we designed our system in a way that can serve the target user in a best way. First, we make assumptions that User will prefer choosing their college Friends as guard when using this app, rather than remote families and friends. Since both User and Friends are very familiar with places around campus, Friends will easily locate and find User when they receive alerts. This will make sure that User can receive the best possible help. Second, we narrow down the use case, by providing only "Alerts" and "Check-ins" features. Therefore, our app is simple to use. Under unsafe situations, the simplicity will save User time sending alerts and get help. Finally, we use GCM instead of SMS messaging to minimize delay. This is essential for a safety app, since timeliness will determine whether the app can provide protect or not.

V. Future Work

Currently, FriendGuard is in prototype stage. More favorable features need to be added, so it provide more secure guarding service. Followings are aspects remain to be improved.

- Enable indoor location sharing. This is to ensure that wherever the users go, their locations can be found and updated precisely.
- One common issue of safety app is battery life. To save battery, in-app computation should be minimized. Also, when mobile device is out of battery, we need to report to the friend before the system shuts down.
- Enable friend selection. Currently, all those on the mobile contact list are regarded as friends. While further work needs to customize the friend selection, so users feel comfortable sharing their information.
- Further user survey. It will help us select the most in-need features.

Acknowledgment

I would like to thank Professor Ashok Agrawala for advising this scholar paper. He provided an interesting and involving course, where this project started and grew. He spent his valuable time guiding me how to read, understand and write paper. I thank Ryan Eckenrod, Master student at University of Maryland, for his contributions on FriendGuard overall design and FriendGuard Server implementations. I also thank Seokbin Kang, PhD student at University of Maryland, for designing the FriendGuard UI.

REFERENCES

[1] Tanya Somanader. President obama launches the "it's on us" campaign to end sexual assault on campus. 2014.

[2] Brian A Reaves. Campus law enforcement, 2011–12. *Bureau of Justice Statistics Special Report. NCJ*, 248028, 2015.

[3] Association of American Universities. Combating sexual assault and misconduct. 2017.

[4] Christopher P Krebs, Christine H Lindquist, Tara D Warner, Bonnie S Fisher, and Sandra L Martin. The campus sexual assault (csa) study: Final report. *Washington, DC: National Institute of Justice, US Department of Justice*, 2007.

[5] Lynn Langton and Sofi Sinozich. Rape and sexual assault among college-age females, 1995-2013. 2014.

[6] Bonnie S Fisher, Francis T Cullen, and Michael G Turner. The sexual victimization of college women. research report. 2000.

[7] David Cantor, Bonnie Fisher, Susan Chibnall, Reanne Townsend, Hyunshik Lee, Carol Bruce, and Gail Thomas. Report on the aau campus climate survey on sexual assault and sexual misconduct. *Association of American Universities*, 21, 2015.

[8] Rena Bivens and Amy Adele Hasinoff. Rape: is there an app for that? an empirical analysis of the features of anti-rape apps. *Information, Communication & Society*, pages 1–18, 2017.

[9] Companion. Companion. https://www.companionapp.io/, 2017. Accessed: 2017-05-16.

[10] bSafe. bsafe. http://getbsafe.com/, 2017. Accessed: 2017-05-16.

[11] DrunkMode. Drunkmode. http://www.drunkmode.org/, 2017. Accessed: 2017-05-16.

[12] Gregory Abowd, Anind Dey, Peter Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.