**ABSTRACT**

We present myStatus, a mobile health app allowing clinicians to collect self-reported health data from patients without compromising patient privacy. While some solutions exist for collecting longitudinal health data from patients, they are cumbersome and require significant patient involvement. In patients dealing with clinical depression, severe problems can develop in the 2-8 weeks that pass between appointments. If clinicians could detect these problems as they happen instead of at the appointment (when they have already become a severe issue), recovery times may improve. The application presented in this paper provides a much easier to use and less intrusive interface for patients, while also protecting private information. The long-term vision of the project is a general mobile health solution useful for more than just depression, which allows for a stronger line of communication between patient and clinician.

# **INTRODUCTION**

Our application started out as a method for depression care. Among patients who have sought depression care, only 50% of those referred to a mental health professional follow through with that referral. Many do not have more than one visit and 50-70% of patients do not have a full response to the first treatment plan [ref]. A significant part of treatment is that the patient has to be willing to participate but part of the reason for unwillingness is that the effort is too great when patients aren’t motivated. Our application seeks to enforce patient self-monitoring in a way that’s not intrusive to everyday activities and does not take an excessive amount of effort.

The Patient Health Questionnaire (PHQ-9) [ref] is a nine questioned survey used by clinicians to monitor the severity of depression. To help clinicians be effective in their treatment, they request their patients to fill these out. Instead of filling them out only when a patient comes in for an appointment, she can have patients fill it out through our application. This then allows immediate help for managing symptoms or finding support.

Because we want to support communication between clinicians and their patients, the application can be extendable. Clinicians should be able to write their own surveys.

Another problem we are addressing is security. Those seeking medical assistance may not want others to know they are doing so. We also cannot have their private information leak out into unwanted hands.

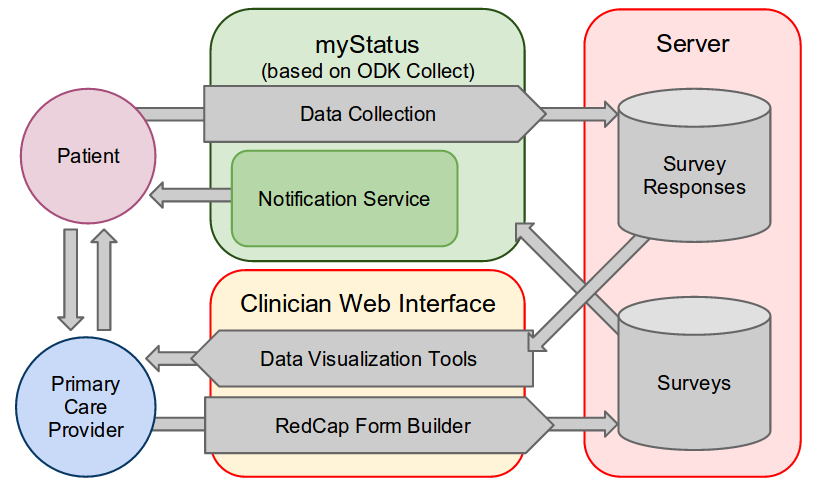
# **RELATED WORK**

In the realm of mobile health monitoring, a lot of applications have to deal with the physical state of a person such as their heart rate or body weight. Then when a user wants to monitor her mental health, options only focus on one form of mental health issue.

“PTSD Coach” [ref]allows users to assess their state through one seventeen question survey. By taking that survey over a period time, a history is developed that the user can track. The user can manage symptoms, but this includes only a set of eight, all of which direct the user to the same sort of exercises in self-management such as deep breathing. Another feature it includes is finding support, either from crisis resources, personal contacts or professional care. They can also learn more about PTSD. One major drawback is that anybody looking through a user’s applications can see that the user is monitoring her depression, which could possibly lead to the user being harmed. PTSD Coach also does not support communication between patients and their medical provider.

Other mental health monitoring applications include headbands that monitor a user’s EEG connected to his mobile device. This can be cumbersome to a patient and we want to give the user the least amount of reasons not to use our application. We want it to be subtle and quick but also provide aid when needed.

# **APPROACH**

**Architecture of Data Collection and Physician Interface**

- Architecture

- Data Collection

- Use XForms to specify surveys

- Collect data using ODK Collect components

- Notify user using language that is not obviously health-related (for

privacy reasons) when surveys are due to be completed

- Submit data to a server such as RedCap or ODK Aggregate

- Physician interface

- View and use visualization tools on data collected from patients

- Currently ODK Aggregate

- May use RedCap if possible

- Build and revise surveys for patients

- Currently ODK Build or XLSForm

- May use RedCap form builder if possible

- Medication Reminders

- Add medication type, quantity, time(s) taken per day

- Allow user to take photo of pill to be displayed with reminders

- Generate reminder using language that is not obviously health-related

when medication needs to be taken. Users can enter the myStatus app

(through the lock screen) to get more detail about what they should

be taking, or dismiss the reminder if they can remember.

- Other apps exist which provide this functionality, but none we know of

provide a privacy barrier--in some, pill reminders can pop up on the

phone's lock screen, informing anyone about the user's medication

regimen

- Activity Scheduling

- Proposed (currently unimplemented) feature particularly useful for

patients with clinical depression

- Suggest activities and schedule on a myStatus-specific calendar

(which may be private if necessary)

- (Optionally) generate reminder before the activity is to take place

- Generate survey some time after the activity is scheduled to end,

asking whether the user managed to do the activity and how it went

- Provide "troubleshooting" if the user was unsuccessful; for each

suggested activity, there should be fallback versions which are more

attainable (e.g., walk to the end of the block and back instead of

walk to the store).

- Most of the preceding components would be reusable as an independent

app; their aggregation into myStatus is mostly useful because of the

common lock screen (since they all expose private health-related

information about the user)

- Data collection encryption would be useful in ODK Collect

- Other components we used (and why we chose them)

- ODK Collect

- Open source data collection tool for Android

- Existing data aggregation and visualization tools

- Support for complex, branching forms in XForm format

- Extensions to and adaptations of ODK could be immediately useful to

other projects

- Limitation: targeted at researchers; not designed for longitudinal use

by study participants

- RedCap

- Existing data aggregation and visualization tools targeted at

clinicians performing surveys

- Support for longitudinal surveys

- HIPAA compliant

- RedCap instance hosted on UWMC servers

- Limitations: data collection is cumbersome on Android, needs to run in a

web browser, notifies longitudinal study participants via email; not

really appropriate for more fine-grained data collection

- SQLCipher

- Easy-to-use database encryption tools

- Limitation: not all ODK data is stored in databases

- SpongyCastle

- Repackage of Bouncy Castle Java cryptography API for Android

- Needed to encrypt files not stored in database

- CacheWord

- Safe method of caching user password for access to private key and

data decryption

- Provides lock screen, locks after inactivity timeout

Security:

Due to the fact that myStatus contains sensitive information, security is a major concern. myStatus is an extended version of ODK Collect which stores data under two main forms: sqlite database and Xforms in the sdcard storage of the Android device. It means that if anyone with the micro usb cable and ADB (android debugging bridge) installed in their computer can expose all the data in the storage of the Android device as long as they know some fundamental linux and sqlite3 commands. On the other hand, MyStatus should also have better protection on the mobile site so that other user rather than the owner cannot easily gain access to the secretive information. Therefore, we need some mechanisms to protect data both from being exposed through ADB as well as from the front end of the devices. Because none of our team members are an expert in data encryption and security, we decided to make use of an open-source library provided by security specialist.

We used SQLCipher [ref], an open-source library, which provides transparent 256-bit AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode. With this feature, anyone who has access to the device through ADB would no longer be able to see any information from the database using sqlite3 commands. However, in order to protect user information in a more secure way, we need a secret key, which can be derived through a user login password. We decided to use the CacheWord library provided by the Guardian project for maintaining and providing the secret key derived from a user's login password. The key features of this library are strong key deprivation using PKDF2 (Password-Based Key Derivation Function 2) and highly secured storage using AES-256. Moreover, this library also provides a configurable timeout which is used by myStatus to lock the app after a certain amount of idle time, thus the user is prompted to type in their password again. Lastly, myStatus uses the SpongyCastle [ref] library for encrypting media files as well as XML files. The SpongyCastle library can make use of the derived secret keys provided to it by the CacheWord service to encrypt and decrypt files. Even if a curious user has access to the Android devices and uses ADB pull to get the media and xml files, they cannot see, watch, or listen to them at all since the files are marked as encrypted.

The time for encryption and decryption can be lengthy for large files and each file must be decrypted before being used, so we follow the approach to create a temporary folder for storing temporary decrypted files which can be reused for a certain amount of time. When the timeout is signalled from the CacheWord service, all the data in the temporary folder will be wiped out before the application lock ups. In general, all the stored data from MyStatus are encrypted and require the correct input password to be able to get the derived key for decrypting.

A major part of the user experience is survey taking. The user needs to take surveys so that myStatus can give valuable feedback. We expect that most users won't go out of their way to do so without a prompt. Our solution was to create notifications that remind the user to do so. To make sure the app reminds them at an appropriate time, we allow them to set which days and at what time is best. Even if the notification appears at what was deemed by the user to be an appropriate time, there may be instances when the user is busy or just not interested. Thus we allow the user to dismiss or snooze the reminder.

Patients with prescriptions fall into three categories in terms of willingness to take them. There are those patients who do not believe medications to be useful, those who want to take them but are prone to forgetting, and those that do so on a regular basis. Our application aims to help the middle group. By allowing the user to enter their prescription information, which includes a picture of the pill, dosage, and how often, we can make notifications to remind them.

# **IMPLEMENTATION**

- Attempted to use ODK Collect as a library

- Impossible to encrypt forms, form media without modifying Collect

- Needed to store additional information about forms; e.g. how much time

should pass before a form needs to be answered again

- No automated support for longitudinal studies; e.g. four distinct

one-shot surveys intended for Week 1, 2, 4, and 8.

- Interfered with normal ODK Collect installations

- Solution: Modify Collect instead, while attempting to make our

modifications as modular, minimal, and well-documented as possible so

that relevant work (like database and form media encryption) could

eventually be used in Collect

- Notification timing

- Goal: Make notifications as unintrusive as possible, allow users to

modify their frequency--don't want to annoy users to the point where

they disable notifications or uninstall the app

- Early on, decided that notifications need to be generated only when

surveys are due (the concept of when surveys become due was a separate

issue)

- Considered triggered notifications as in myExperience [ref], but triggers

available to the phone are either not relevant to health, or can be

adapted to the lazy-evaluation model.

- Considered several models for notification timing

- Allow users to set notification times like Google Calendar events,

setting arbitrary dates and times when it would be convenient to

receive notifications

- Too complicated

- Allow users to set several notification times per day of the week,

e.g. before/after work on weekdays and mid-day on weekends

- Still too complicated. Ultimately we decided that having surveys

due at multiple times per day was going to be too annoying for most

people and unlikely to actually produce valuable information.

- Allow users to set what time of day and which days of the week they

would like to receive notifications.

- Having a single time of day greatly simplifies the notification

timing interface. This has the disadvantage of requiring that

notifications always arrive at the same time, but we decided that

having a "snooze" button on notifications was a sufficient

compromise.

- Due surveys evaluation

- Originally planned to allow arbitrarily complex logical predicates

- e.g. "hoursSinceLastResponse(2) and (inTimeRange(8:00-12:00) or

inTimeRange(16:00-20:00))"

- Didn't fit well with the lazy evaluation model--what if notifications

are never generated within the range when the predicate is true?

- More complicated than necessary

- Went with a simpler model where only a few conditions can be specified:

- Days between responses to a periodic form (e.g. PHQ-9 should be

responded to every 14 days)

- One-off forms (first time setup, feedback surveys)

- Forms with a time delay (downloading a longitudinal survey package

with four forms set to appear after 1, 2, 4, and 8 weeks)

# **EVALUATION**

- Difficult to test in a "real-world" setting due to privacy concerns and

study restrictions

- Surrogate users are students in usability tests

- Ensure app is easy to use and unintrusive

- Working closely with Amy Bauer to ensure that we provide all needed

features and expose them in a reasonable way

# **CONCLUSION AND FUTURE WORK**

Working through this project helps each of us gain many important experiences especially from programming and teamworking. We have learnt quite a good amount of Android Programming and Mobile Security which are applied straightforwardly to MyStatus. Also, learning how to use Git version control effectively is also an important skill as working in a team because it will enable you to stay in touch with other team members works as well as updating your workspace when necessary. Due to the fact that our schedule is not quite flexible this quarter, we were independently working on different aspects of the project and tried to merge with each other weekly. However, this strategy does not work out really well for us at the beginning because we end up not having clear idea about what other team members working part. We change to set up some more time for coding section and meeting which can help us learn and help each other more effectively (To be continued).

**ACKNOWLEDGMENTS**

# **REFERENCES**

-<http://www.agencymeddirectors.wa.gov/Files/depressoverview.pdf>

-<http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6346977&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6346977>

http://en.wikipedia.org/wiki/Block\_cipher\_mode\_of\_operation#Cipher-block\_chaining\_.28CBC.29