Evaluating performance of UI interactions: Interactive vs non-interactive notifications

Amanda Ma, Selena Nguyen, Harman Dhaliwal, Daivakshi Vaidya
Dept. of Electrical Engineering and Computer Science
York University
Toronto, Ontario, Canada M3J 1P3
harman50@my.yorku.ca, sen1881@my.yorku.ca, amanduhh@my.yorku.ca, daivakshi@gmail.com

ABSTRACT
Mobile computing has led to a demand for intuitive and user-friendly mobile user interfaces. Taking into account power consumption and medication non-adherence, an interactive notification-based reminder system would be better suited for users' needs. It would provide users with correct and accurate information while also being more efficient. This paper discusses the results of a comparative study conducted on a mobile phone and tablet to investigate whether performance and resource consumption were more efficient in an interactive notification than in a non-interactive notification. The experiments consisted of installing the application and adding pills. Then profiling the interactions within the app as well as in the notification buttons to mark the pills as taken. After CPU, energy and memory data were gathered from the profiling results, it was concluded that there were no significant differences in the resource consumption for the notification interactions, therefore, supporting the null hypothesis.

Keywords
Mobile User Interface, Medication, Reminders, Notifications, Performance Comparison.

INTRODUCTION
As mobile computing has grown over the past years, the number of apps have as well. The development of these apps has led to a growing demand for intuitive and user-friendly mobile user interfaces. These designs need to be created so that they are easy on the eyes and as efficient as possible. Developers also have to take users with disabilities and limitations into account. This also leads to the notion of power consumption. Mobile apps have continued to evolve and gain more appealing graphics and animations, but this has also caused the battery to be consumed more quickly. While trying to make a well-designed app with all these features, designers still need to take into account how to optimize their code. This is to ensure phones have a long-lasting battery that meets users' standards. Overall, creating a well-organized design is challenging in its own right, and when mixed with the idea of conserving battery power, creating a well-designed app is very arduous.

Medication non-adherence is a major issue that can have significant consequences on a person's health and well-being. Many people tend to forget to take medication rather than being physically opposed to the medication itself. This problem is amplified by those who have a chronic neurodegenerative disease like Alzheimer's. Issues that affect one's health can damage the ability to remember important daily activities and can be a hindrance especially when it comes to taking medication. This issue of remembering to take medications not only affects the recipient of the medication but also the stress it causes loved ones who care about them.

A better-designed UI experience that would provide a better experience to the user would help with this problem. A major issue with many reminder apps is that when they send informational notifications, they tend to be ignored, or pushed to the side by other urgent matters. This causes a clutter of notifications. This is why using an interactive notification-based reminder system would be better suited for users' needs. An interactive notification-based reminder system would still provide correct and accurate information so that a user can remember to take their medication, how much of it, and when to take it. The difference here is that the user, who would normally use their phone for other tasks, will have these notification reminders whenever they need to take their medication and they wouldn't have to open the app, go through all the reminders, and check what they need to do that day. Rather the notification can give them the information they need for the specific medication and they can simply interact with a simple button on the notification.

Opening the app-based reminder system directly would also continuously drain the battery since the application has to be in focus. However, the interactive notification-based design will only consume power for the notification alert and button interaction. The battery consumption of the app would be lower than when compared to similar apps. The only times a user would need to open the app would be when they initially set up dates and times of medication, as well as when that information needs to be edited or changed.

The aim for the interactive notification-based design of a medication reminder app was to have a better-implemented UI design by focusing on the part of the phone that people view the most; the lock/notification screen. Not only would it do a better job at reminding its users, but the power consumption would also be lower than the traditional reminder app.

RELATED WORK

Frank Bentley [1] had their participants fill out a survey to investigate the "enabling or disabling [of] notifications on Android and iOS [apps]" [1]. After 2.5 years, participants were asked to fill out the survey again. The survey shows that "82% of Android users had personal mail notifications enabled, only 58% of iOS users did" [1]. The newer survey talks about how social networking and messaging apps became more frequently used, and the number of notifications being sent out daily increased. The results show that "iOS users have turned on notifications much more frequently, Android users have disabled notifications for core apps such as SMS, Email, and Facebook and have increased in other categories at a slower rate" [1]. More iOS users enabled notifications compared to Android users within the past 2.5 years.

Bentley and Tollmar [2] created a health app that asks users about the food they've eaten that day and their overall health. They conducted a study on how implementing notifications through "simple status-bar reminders on a mobile phone can increase self-logging frequency" [2]. It's important to send users notifications to remind them to complete their tasks, since they may forget otherwise. The notifications ask easy questions, measured on a numbered scale. Users could also add times for the notifications to be sent and how many times to repeat the notifications per day. The notifications are more helpful and convenient since they help remind users to log and input information. Bentley and Tollmar recommend that app developers use notifications that should be visible and not disrupt the user with loud sounds. Notifications should have options that make tasks easy to complete.

Chen et al. [3] discuss common issues that users have with mobile user interfaces in their study. They give recommendations for mobile app developers on how to improve the user interface of an app to better suit the needs of their users. One of the most common issues that users have with mobile user interfaces is with notifications. Users expect to receive notifications when there are any changes in the app. Users are often discontent when an app does not send notifications when expected, and even more so if an app sends too many notifications. Apps that send more notifications consume more resources and drain the battery faster as well. The study recommends that app developers should make sure that users receive notifications when they expect them, and to avoid sending out too many notifications. Notifications should mainly be used to update users on any changes in the app, and give important information.

Shirazi et al. [4] conducted a study in order to investigate which apps users allowed notifications from, and which they were more likely to disable notifications for. It talks about how mobile app developers can better design notifications to be more engaging to users and less cumbersome. Users are more likely to interact with notifications if they're engaged with the content or see them as important. They tend to dislike receiving too many notifications from apps that are not as important. The study was conducted by using an app called "Desktop Notification" that sends the notifications a user receives on their mobile device to their desktop PC. The app also has a "blacklist" function to disable notifications for any apps that the user chooses [4]. The results showed that users preferred receiving notifications from social networking and messaging apps, and that they were more likely to disable notifications from system apps and less important apps.

METHOD

Hypothesis Statement

In this comparative study, profiling data such as CPU, memory, and energy consumption was used to investigate if performance and resource consumption were more efficient in an interactive notification than a non-interactive notification that opens the application, in both a phone vs tablet setting.

Apparatus

The experiment was conducted on two alternative device settings; a Google Pixel 5 running Android 9.0 (API 28) Pie, with 6 in. display, 1080x2340 resolution, and 440 ppi; and a Nexus 9 tablet running Android 9.0 (API 28) Pi, with 2040x1536 resolution and xhdpi.

The medicine reminder app was developed in Java using the Android SDK (in Android Studio IDE). When the application is started, the splash page opens and automatically leads to the main page/dashboard. The following use cases were implemented for the application:

The Dashboard contains a scrollable view at the top that shows medicine info for the current day. Users can view which medicine they will have to take at what time of the current day, as well as record whether they have taken the medicine for that time.

Below the scrollable view are 4 interactive buttons; calendar, view medicine, add medicine, and settings.

The add medicine button opens a new page/activity where the user can input info/details about the medicine (name, amount, etc.), and schedule (time of day to take, how many days per week).

The view medicine button on the dashboard opens a new page. It shows the user's current list of medicine they have added and the details of each medication. Users will also be able to edit medicine information. Clicking on the edit button will open a similar view to the add pills page, but users are able to override the current information of a certain medication.

The calendar and settings pages were not implemented as they were irrelevant to the comparative evaluation of the notifications.

Due to difficulties when implementing the scheduling for notifications, the users are notified for each medicine they have added with a 2-minute delay to mock the scheduled notifications. There were two types of notifications implemented. The first contains details of the medicine that they must take at that time and interactive buttons so the user can record whether they take their medicine for that time or skip it without having to open the app. The other notification type is not interactive; the user must open the app to update the information.

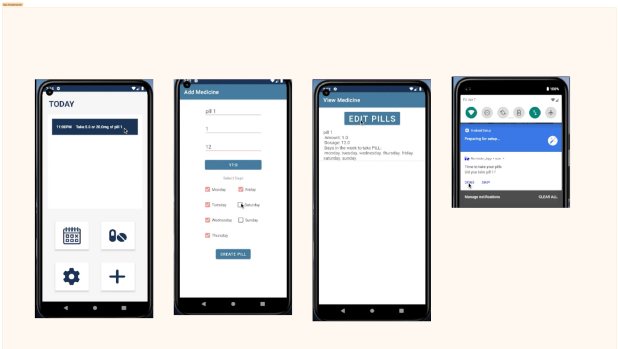

Figure 1. Screenshots of the home, add and view medicine pages of the reminder app, with interactive notification.

The UI interactions and use cases of the app were tested using Android Studio Espresso

The performance evaluation/comparison was conducted using Android Studio profiling tools which included the CPU profiler (track runtime performance issues), Memory profiler (track memory allocations) and Energy profiler (track energy usage).

Procedure

The performance evaluation was conducted on the Pixel 5 and Nexus 9 tablet with comparisons between two UI features - an interactive notification and a notification that opens the application (4 experiments in total). The interactive notification contains information regarding what pill/medication(s) needs to be taken at the time and the dosage amount. Along with this information, the notification has two buttons; done and skip to indicate if the pill/medication(s) has been consumed. The second notification is a notification with text reminding the user about the medication but does not give any information about it or any way to interact without opening the app.

Since the second notification opened the application, the full application UI was launched and the comparison was made with the app running in the background vs running in focus.

Before running the profiler for each test/experiment, the application environment was set up with the following steps:

1. Launch the application with the profiler.
2. Added a pill that is to be taken every day, with a selected time
3. The home button was pressed to have the app running in the background.
4. Start a profiling trace.

The UI interaction during the profiling included waiting to receive the notifications and either interacting with the notification buttons to mark the medication as complete/skipped or clicking the notification to open the application (in focus) and marking the medication as complete/skipped there.

CPU Profiler

This tool was used to track the CPU/computing resource usage of the app in both interactions. The following steps were taken for each respective interaction:

1. The Application was launched and set up with predefined pill data with the steps above.
2. After the profiling trace was started it displayed event, activity and CPU timelines in real time.
3. A recording configuration was to be selected (from the profiler menu) to record the timelines and profile data
4. Started Recording and performing the interaction.
5. Upon completion of the interaction, stop recording.
6. Immediately after stopping, all tracing information that was recorded was displayed
7. Filters, time references, analysis panes, etc., were used to get specific information needed for the side-by-side comparison.

When focused on the CPU timeline in the profile view, the detailed percentage data of the CPU the app was using, the usage of other apps and the total active threads of the app during that time were gathered for comparison.

Memory Profiler

This tool was used to track memory leakage and the app's use of memory in the device. The process was as follows:

1. The Application was launched and set up with predefined pill data with the steps above.
2. After the profiling trace was started, it displayed a detailed timeline of the app's memory use along with other options to track memory allocations
3. The same recording fundamentals as CPU profiling were used to get an allocation record (record Java/Kotlin), which contained a detailed view of allocations for every instance.

Energy Profiler

This tool was used to estimate how much energy/battery was being consumed by different components run by the application. The process was as follows:

1. The Application was launched and set up with predefined pill data with the steps above.
2. After the profiling trace was started, it displayed a detailed timeline for system events, estimated energy consumption of the app and events throughout the interaction, in a bar chart/graph format.
3. Changes in these timelines throughout the interaction are kept note of, and the results were recorded on a table (Eg. average light, medium or heavy consumption for each timeline along with the count of events including system events)

RESULTS AND DISCUSSION

Results

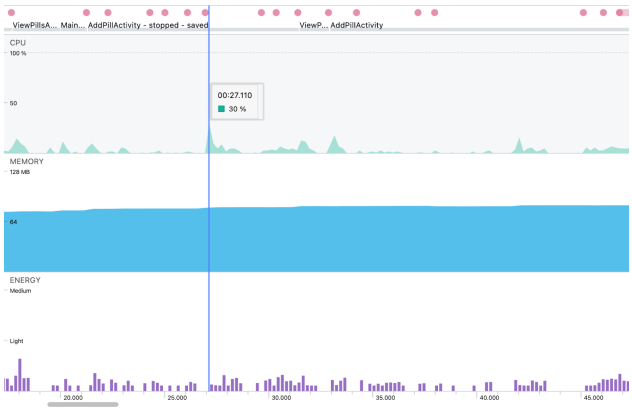

Figure 2. Profiling screenshot in regards to CPU, Memory and Energy consumption for interactive notification on mobile.

As shown in Figure 2, the profiling started with the main activity page and constant resource usage in memory. At around the 27-second mark, we added a pill/medication, which caused a spike in the amount of resources consumed. After the short spike, the CPU usage went down as the pill information was processed and stored, which returned the resources to a stable position again. These spikes were very frequent and the app frequently rose to nearly 30% CPU usage for the interactive notification on mobile.

The energy usage of the app was minimal. As per figure 2, it is shown that the app didn't pass the "light" indication in the energy profiler throughout the entire profiling session.

The memory used by the app was consistently slightly more than 64 MB, which only rose further as pills were added and the profiling continued.

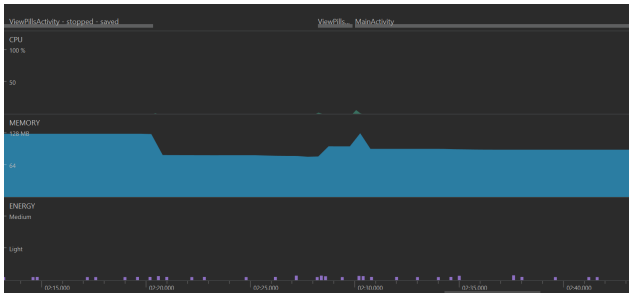

Figure 3. Profiling screenshot in regards to CPU, Memory and Energy consumption for a non-interactive notification with a mobile phone (Google Pixel 5).

Figure 3 shows the profiling for CPU usage for a non-interactive notification for a phone. We can see that the jumps added when adding a pill were not as much compared to the interactive model. The memory usage for the non-interactive started off much higher than for the interactive model taking more resources from the moment the application was launched.

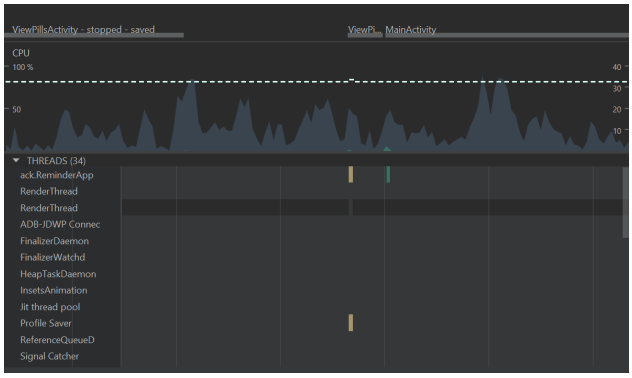

Figure 4. Profiling screenshot in regards to CPU usage for a non-interactive notification with a mobile phone (Google Pixel 5).

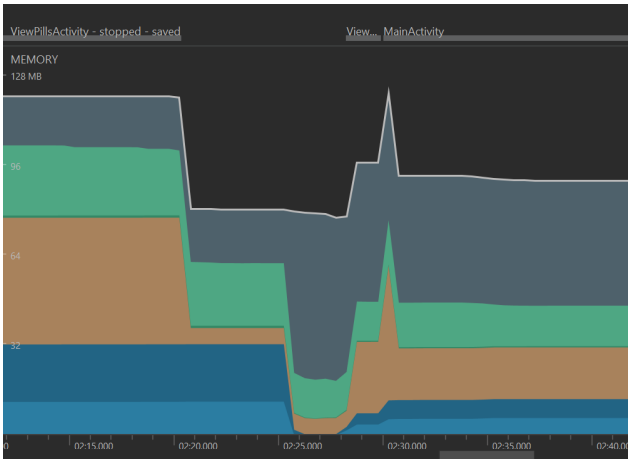

Figure 5. Profiling screenshot in regards to Memory consumption for a non-interactive notification with a mobile phone (Google Pixel 5).

In Figure 5, the profiling screenshot in regard to memory consumption for non-interactive is seen in more detail. The initial value of memory usage is well above the 64 MB recorded for interactive. The memory value was nearly double for the non-interactive version, and even after going down remained over 80 MB.

Figure 6. Profiling screenshot in regards to Energy consumption for a non-interactive notification with a mobile phone (Google Pixel 5).

In Figure 6, the screenshot of energy consumption in non-interactive notifications it is evident that power consumption was very minimal. There were multiple spikes when actions were performed but always remained far under the light usage indicator.

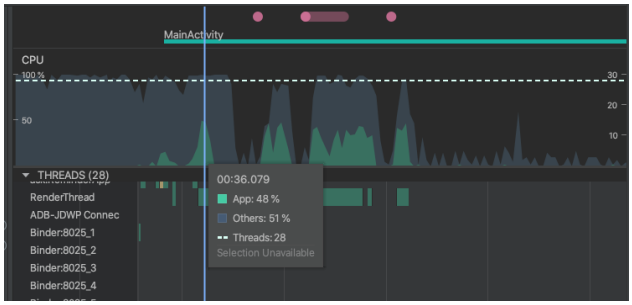

Figure 7. Profiling screenshot in regards to CPU usage for a non-interactive notification with a Nexus 9 tablet

Figure 7 shows that the CPU usage of the app ranged from 31 to 48%. There are peaks in CPU usage when adding a pill and receiving a non-interactive notification.

Figure 8. Profiling screenshot in regards to Memory consumption for a non-interactive notification with a Nexus 9 tablet

Figure 8 shows that the memory consumption for a non-interactive notification was usually under 128 MB. There's an initial spike that goes over 128 MB, but after that, consistently stays around 64 MB and under.

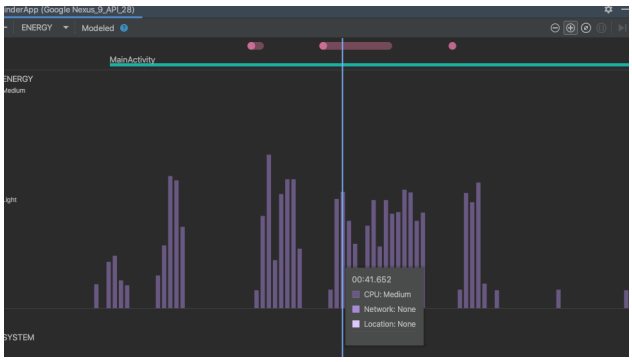

Figure 9. Profiling screenshot in regards to Energy consumption for a non-interactive notification with a Nexus 9 tablet

Figure 9 shows that the energy consumption for a non-interactive notification with a tablet was considerable. The CPU was at medium energy usage.

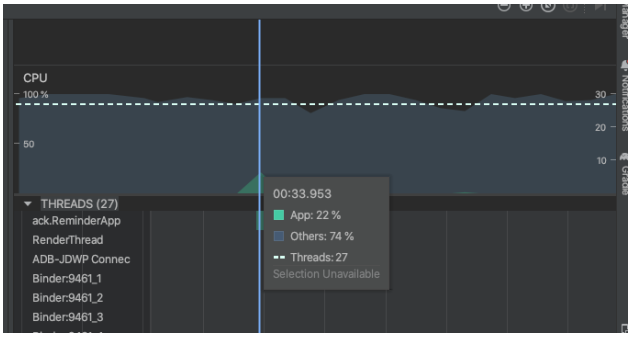

Figure 10. Profiling screenshot in regards to CPU usage for an interactive notification with a Nexus 9 tablet

In Figure 10, it shows that the CPU usage of the app is slightly lower with interactive notifications. The CPU usage of the app with interactive notifications was 22%, compared to non-interactive notifications, where the CPU usage was around 31 to 48%.

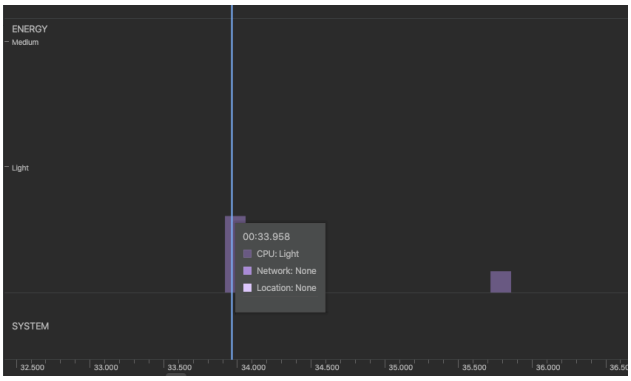

Figure 11. Profiling screenshot in regards to Energy usage for an interactive notification with a Nexus 9 tablet

Figure 11 shows that the energy consumption is slightly lower with interactive notifications. With interactive notifications, the CPU was at light energy usage, while with non-interactive notifications, the CPU was at medium energy usage.

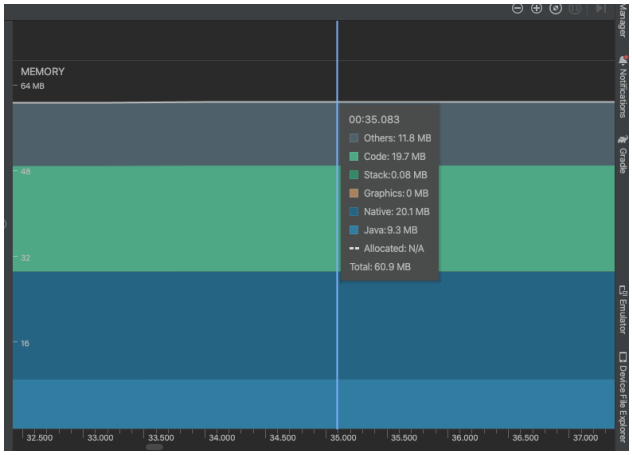

Figure 12. Profiling screenshot in regards to Memory consumption for an interactive notification with a Nexus 9 tablet

In Figure 12, the memory usage stayed under 64 MB. This shows that memory usage between interactive and non-interactive notifications is roughly about the same.

Discussion

Overall it can be seen that memory usage is higher when using the non-interactive notification that opens up the application, for both mobile and tablet settings.

For CPU consumption, the results for mobile showed that the non-interactive notifications took less CPU, whereas the tablet results showed that CPU consumption was less with the interactive notifications.

With Energy, there was only a significant difference in the tablet results, showing that the interactive notifications used light (as compared to non-interactive at medium usage). In the mobile setting, energy results were very similar with both interactions.

Subsequently, these differences in resource consumption between the mobile and tablet settings were not enough to conclude that there was a significant difference in using interactive or non-interactive notifications. Therefore, it supports the null hypothesis.

CONCLUSION

Here we discussed the development of an interactive notification-based medication reminder system as a solution to the problem of medication non-adherence. The system was designed to be more user-friendly and intuitive while also taking into account power consumption. A comparative study was conducted to investigate whether performance and resource consumption were more efficient in an interactive notification than in a non-interactive notification. After analyzing the profiling results of CPU, energy, and memory data, it was found that there was no significant decrease in resource consumption for the interactive notifications, thus supporting the null hypothesis. This study contributes to the development of a better-designed UI experience that provides a better experience to users and helps them remember to take their medication. The interactive notification-based reminder system is a more effective way of reminding users, and it consumes less power than traditional reminder apps.

REFERENCES

1. Bentley, F. (2018). Losing the Power of Defaults. Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers. https://doi.org/10.1145/3267305.3274109

2. Bentley, F., & Tollmar, K. (2013). The power of mobile notifications to increase wellbeing logging behavior. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. https://doi.org/10.1145/2470654.2466140

3. Chen, Q., Chen, C., Hassan, S., Xing, Z., Xia, X., & Hassan, A. E. (2021). How Should I Improve the UI of My App? ACM Transactions on Software Engineering and Methodology, 30(3), 1–38. https://doi.org/10.1145/3447808

4. Sahami Shirazi, A., Henze, N., Dingler, T., Pielot, M., Weber, D., & Schmidt, A. (2014). Large-scale assessment of mobile notifications. Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems - CHI '14. https://doi.org/10.1145/2556288.2557189