Where Did The Time Go?

John Viaud, Bilal, Vitali, Dr. Jia Yanxia
Arcadia University, Glenside PA

INTRODUCTION

Study shows that most people spend a full quarter of their active hours on their mobile device, limiting productivity and potentially taking a toll on their health. Our Android application is able to track and visualize device usage to help users understand how much they use their devices.

At the start of building this application Android Digital WellBeing was not available to the public and iOS Screen Time had recently been released. So to separate ourselves from these other apps we tried to focus on what made those two apps great and where they could use some improvements.

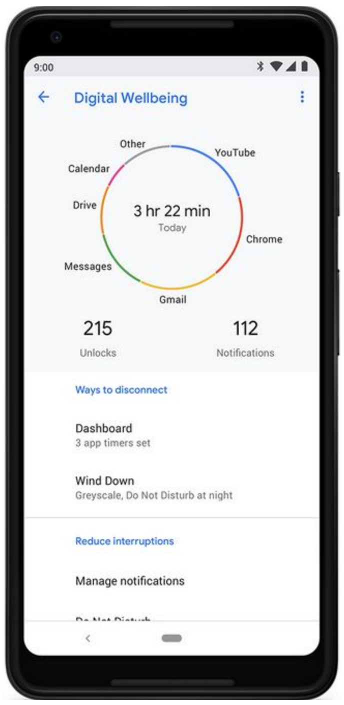
As a result our application focuses heavily on the user interface and user experience as well as personalizing the data being presented by allowing users to set goals that are projected over the data we collect.

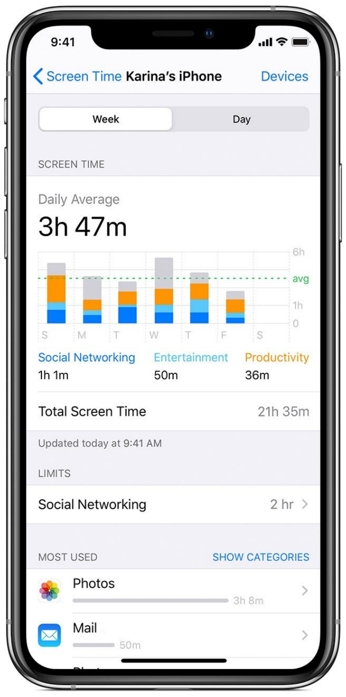
MOTIVATION

Incoming college students often face difficulty with time management because they spend excessive amounts of unnecessary time on their cell phones. We were motivated to create a user friendly app that will help users better manage their time. We want people to have an app that allows them to track how often they pick up their phone as well as opens specific apps. This app will help people use their time wisely. This app is useful in time management for anyone but particularly for new students who have to adjust to their new schedule and classes. Many people often spend a lot of time on some apps and they may be unaware of how long they are on it. Users will be able to set goals for themselves for the amount of time they want to spend on certain apps. We also wanted to improve the functionalities of Apple’s Screen time and Androids Digital wellbeing apps which are similar to our app.

USER INTERFACE / USER EXPERIENCE

Navigating iOS' Screen Time can be exhausting, at times I find myself having to relearn how to access certain features or being completely unaware a feature exists. As a result, even though Screen Time has an excellent design, the user experience leaves much to be desired. Alternatively, Android's Digital Wellbeing has a simple navigation but an even simpler and dated UI.
When designing our application, we aimed for a modern look that would best display the large amounts of data we are collecting from our users without overwhelming them. This means that pages must be sophisticated to a degree that does not confuse users but simplistic enough to be intuitive. To achieve this we followed Google's Material Design guidelines while also looking for inspiration from other apps.





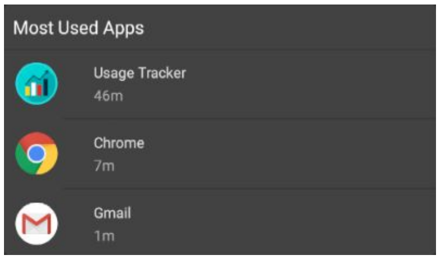
Android Digital Wellbeing

iOS Screen Time

Color

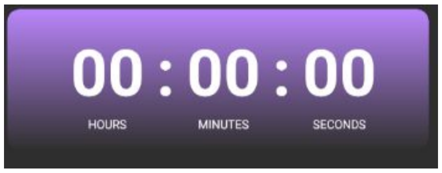
One of the most important points when designing a UI is color. We can use color to create depth and draw the users attention to places we deem to be important. It is also important to establish a theme, which is a set of colors that will be consistently used throughout our app to make our application feel more harmonious and help users become more familiar with our app.

Color, like size, can also be used to represent a hierarchy. As you can see in the example below, There is a clear hierarchy in titles. The first title has a value of white (#FFFFFF), then the titles within the list view also have a value of white but has a reduced opacity of 87% (#DEFFFFFF) and finally its content has a further reduced opacity of 60% (#99FFFFFF)

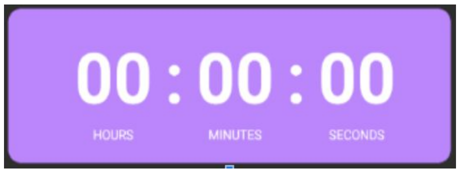


Color 1

Gradient



Gradient 1

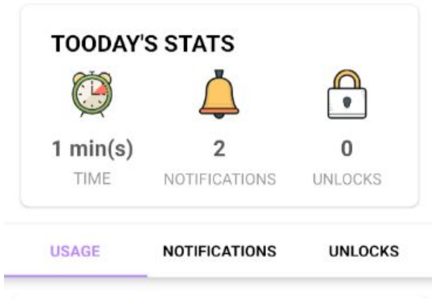


Gradient 2

Here we use a gradient effect to give a cardview the illusion of depth. In the examples above, both images have the same color purple but the second image appears flat whereas the first has the illusion of depth and draws the users eyes, telling them that this is an area that requires attention.

Elevation

Elevation also allows us to directly create depth by putting a shadow behind our content, the darker the shadow the higher the elevation. This lets the user know that this particular content with an elevation is a potential point of interest.



Elevation 1

Here we direct the user to useful information like today's statistics and a tabbed navigation.

SYSTEM ARCHITECTURE

At the core of our app is Android's Broadcast Receiver which is a system service built into the Android operating system that will wake and notify our application when a designated event occurs.

Notification Listener Service

- Notification Listener is a Broadcast Receiver that sends alerts to our application whenever a notification is received by the system.
- Due to security measures implemented by Google, restrictions have been made to disallow certain Broadcast Receivers from being generically created in the Android Manifest allowing the service to start our app to collect data from the Broadcast Receiver.
- To go around this we start and register this Receiver in a Background Service to allow us to always run and send data to our application.
- Similar to the Usage Stat Manager, Notification Listener is a protected permission so we can not conduct a self check but we can start the activity to allow the user to give our application the correct permission. We then check to see if the Listener has been connected, which is a default method that is triggered when the Listener successfully starts.

Unlock Listener

- Unlock Listener is a custom Class that extends Broadcast Receiver and is registered with an IntentFilter of ACTION_USER_PRESENT.
- This receivers is triggered whenever a user unlocks their device

App Change Listener

- App change Listener is a custom Class that extends Broadcast Receiver and is registered with an IntentFilter of ACTION_PACKAGE_ADDED and ACTION_PACKAGE_REMOVED
- Because we keep a list of all other applications installed on the device we need to be notified when there are changes. This receiver is triggered whenever a user adds or removes an app allowing us to only reconstruct the list when necessary.

Background Service

Android offers applications the ability to run in the background as a service. This is useful for us for a couple of reasons. The only thing that is required is a sticky notification letting the user know our app is running.

- Because we are polling usage data our app must be active to run the polling, the background service allows our app to run silently in the background as it continues to poll for usage data.
- Due to security concerns, Android had limited the ability of broadcast receivers only allowing certain receivers to wake apps making others only available while the registered application is running. The broadcast receivers used by our application are not exempt from these restrictions but since our application is always running we can get around this.

Usage Stats Manager

- USM is a system service that polls the system for apps in use. It gathers basic information like the package name of the app in question and usage in milliseconds over a specific period in time.
- USM stores this data in a HashMap<String,UsageStats> where the Key is the package name of the app and the value is a UsageStats object.
- ***UsageStats***
    - This object allows us to conduct basic queries such as getting the corresponding application's package name, total usage for the specified period of time and the #getLastTimeUsed() which returns a long.
- Because this permission is protected by Android, non system apps can not properly check for this permission. The activity to receive the permission can be started but a standard self check will always return false regardless if the app has the permission.
- To go around this we create an Object of USM and try to query data for our own application for the last 1 minute, if the query is unable to return data then we assume we do not have the permission.

Polling

Originally we were set out to use USM to poll all usage data whenever the user starts our app since USM is capable of gathering data from any given time period. However, due to a bug in USM, it ignores the period of time we specify and provides us with incorrect data. Because we can no longer get the usage data retroactively we must continuously collect the data ourselves

<u>Manual Polling (Usage)</u>

- Since our application is always running in the background we can manually collect and store usage data.
- Android's Package Manager has the ability to check what apps are currently running but these methods have since been deprecated due to security concerns. Alternatively, USM works well enough that it always returns the most recent usage data. We can not cherry pick time intervals within this collection but

since we know this is always the most recent data we can order them in a SortedMap by the last time they were used.

- It will be extremely inefficient and resource consuming for our application to collect and store data every 1 second since we have to sort a Map whose size we cannot guarantee then add to a database. We can assume there will be too much collision even in a background thread. To combat this we only poll every 1 minute.
    - Every minute we check to see what was the last app used at the end of that minute and assume it was used for that whole minute.
    - Points of Error
        - If multiple apps were used with this minute then only the last app is recorded
        - If an app is opened at exactly the 59th second then it will be recorded as being used for the whole minute
    - Because of these errors the usage being reported has an error of about +/- 59 seconds

Manual Polling (Unlocks + First App Used)

- Android provides easy methods to check if the the device is locked with KeyGuardManager
- We can also see if the device is unlocked by registering a BroadCast Receiver with an IntentFilter of ACTION_USER_PRESENT which is triggered whenever the device is unlocked.
- From here we can use the same method as tracking usage to see what is the first app used during the period from when the phone was locked and then unlocked.

Manual Polling (Notifications)

- Notifications do not need to be manually polled. The Notification Listener will notify our application of any notifications.

Database (SQLite)

Usage Table

ENTRY_ID	USAGE_DATE	HOUR_OF_DAY	PACKAGE_NAME	UNLOCKS_COUNT	NOTIFICATIONS_COUNT	USAGE_TIME	APP_CATEGORY

ENTRY ID:
STRING, PK, combination of package name date and hour of day
REQUIRED BECAUSE IT IS USED TO PURPOSELY THROW SQLConstraintException to check if an app has already been inserted.
USAGE_DATE: STRING, Date the usage represents, must be in yyyy-MM-dd format
HOUR_OF_DAY: INTEGER, The hour of the current day from 0-23

PACKAGE_NAME: STRING, The name of the package that the current stats belong to
UNLOCKS_COUNT: INTEGER, The number of times the app was the first one opened after an unlock event
NOTIFICATIONS_COUNT:INT, The number of notifications the app has received
USAGE_TIME: REAL (LONG), The amount of time the app has been used in milliseconds
APP_CATEGORY: The category a particular app belongs to as determined by the Android OS

Goal Table
ENTRY_ID: INTEGER PRIMARY KEY AUTO INCREMENT
GOAL_DATE: STRING, Date of goal in yyyy-MM-dd format
GOAL_TYPE: STRING, Goals can either be a “PHONE_GOAL” or an “APP_GOAL”
GOAL_USAGE:REAL (LONG), The amount of time specified for the usage goal
GOAL_UNLOCKS: INTEGER, The number of unlocks specified
PACKAGE_NAME: STRING, The package name of the app specified if applicable
GOAL_STATUS:INTEGER, The status of the overall goal (Pass or Fail)
USAGE_STATUS: INTEGER, The status of the usage goal (Pass or Fail)
UNLOCK_STATUS: INTEGER, The status of the unlock goal (Pass or Fail)

Attribute References

By giving views references to colors and styles instead of hardcoding them, we can easily change the value of the references in a single location to have the changes take effect across the entire app. We take advantage of this when implementing the dark mode feature to quickly change the colors of all the views across the application when the user changes between light and dark mode.

FEATURES

Goals

Users have the ability to set goals for themselves, this is different from the limits in that the goals are not enforced. Goals are merely superimposed onto the usage data that we are already collecting so users can have a better understanding of how much they use their device and to personalize the data.
Since users do not have any control over what application sends them a notification, goals are limited to only usage time and number of unlocks.

Settings page
On the settings page there are general settings, parental controls, and advanced settings.
General
In the general settings, users can select which apps they want to track on the phone by selecting an app, if the user wants to stop tracking an app on the phone, users can deselect an app which completely removes that app from the database so it'll no longer be tracked.
Dark Mode
Dark mode allows users to change the theme of the app from light to dark. This will completely change the color display on all parts of our app.
Parental Controls
In the parental control setting of our app, users can set two limits for their phone. One for phone limits and one for app limits.
Phone Limit
If a user clicks on the phone limit, they can set a specific phone limit for the total usage of their phone. This will be shown by the timer on the homepage that is counting down from the limit that was set. Once this limit reaches 0, the user will receive a notification that says your phone usage for today has been reached. So you can no longer use apps on the phone. If you try to still use apps on the phone the notification will pop back up and once you press okay, you'll be redirected back to our app.
App Limits
For app limits, users can select which apps they'd like to limit for the day. Once they set a time limit for an app, it'll be displayed in time limit apps.
Wifi
Users can turn wifi access on or off. One Wifi access is turned off the user will not be able to connect to the internet via wifi unless this setting is turned off from the application
Advanced Settings
For the Advanced settings, users can export to csv which allows users to view a usage report of specified data from the app.

LIMITATIONS

Certain functions of the application are limited due to security restrictions put in place by Android. For example there are a number of Broadcast Receivers, Permissions and services that are System Protected and can only be accessed by a system application.

There is also the issue of certain methods that would have made our application more efficient being deprecated.

LIBRARIES

Hello Charts
Lottie
InkPageIndicator
MaterialFilePicker
OpenCSV

FUTURE WORK

- Improve efficiency of application
- Expand upon the Goals functions
- Add the option to import data
- Maybe store data on the cloud so users can access their usage information from anywhere (like a Webpage)