Tundra: Study App

JR Ansera
University of Chicago
Chicago, Illinois, USA

Francis Yang
University of Chicago
Chicago, Illinois, USA

Marc Maliar
University of Chicago
Chicago, Illinois, USA

ABSTRACT

Since the rise of mobile computing, the smartphone has become a key component in our day-to-day life. However, its role as a facilitator in social media, the internet, media consumption, and gaming make smartphones incredibly distracting and threaten to ruin productive study time. *Tundra* is a study app built on Android for Java that disables your phone while you are studying. To study, the user must flip the phone upside down and is not allowed to flip the phone back up to look at it during the duration of the study time. By completing the study time block, the user gets rewarded through the app and they rise in the global study rankings. To the best of our knowledge, *Tundra*'s distraction-blocking mechanism is much more sophisticated, thorough, satisfying, and helpful than that of any other competing study app—*Tundra* promises to revolutionize the study app marketplace.

CCS CONCEPTS

• **Applied computing** → **Learning management systems**; • **Hardware** → *Power estimation and optimization.*

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
JR Ansera, Francis Yang, and Marc Maliar. 2022. Tundra: Study App. In *Proceedings of UChicago Mobile Computing Group (UCHI).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

During the COVID-19 pandemic, many of us found ourselves glued to our screens for many hours at a time for the purpose of school or work. However, rather than studying, we found ourselves getting distracted with social media or games. What we really need to ensure productive study periods is an effective study app.

To use a study app, the user opens it, sets a timer, and works while the timer is running. This block of time is a special time that the user has set aside to completely focus on study. And so a great study app complements the user's efforts to completely focus on the work. For example, the app may remove distractions—disable the internet, certain apps, websites, and silence phone notifications. It can also try to help the user come up with a strict study schedule so studying becomes more natural. Finally, it can also offer an extra incentive for the user to study—for example, studying can be made into a game in which you compete with your friends for who studies more.

As students, we have not been satisfied with the current state of study apps. So, we have decided to build our own study app *Tundra.* To our knowledge, *Tundra* is the only app that fits our requirements. Here are a few key features of *Tundra.*

- *Tundra* completely eliminates phone distractions during the study block of time. When you set the timer, *Tundra* asks the user to flip the phone upside down. If the user tries the flip the phone up to use it, *Tundra* will stop the timer and the user will not get a reward for completing the study time.
- *Tundra* encourages the user to forget the phone completely. By flipping the phone upside down, the phone is effectively out of sight and out of mind. The user can really focus on studying.
- *Tundra* sets a user against their friends for who can study the most. This social reward provides an incentive and (in our experience) does not interfere with focused studying.

We have evaluated *Tundra* on several Android phones, including the Samsung Galaxy S21 and Oneplus 7T Pro. We have concluded that the *Tundra* user experience is effortless and positive. The app's flipping mechanic is in particular very stable across a variety of test situations. With a few more quality-of-life changes, *Tundra* has the potential to revolutionize the study app market. Studying has never been easier.

2 RELATED WORK

The *Forest*[**?**] is a popular study app which has been extensively used by one of our group members. In the app, a user sets a study timer as is usually done in study apps and tries to study during the timer. If they make it all the way through a virtual tree is "planted" in their app. On the other hand, if the user leaves the app before the timer ends, the tree withers. A user accumulates trees in their virtual forest as they study day by day.

Our group member was left unsatisfied with the app. First, it does not do enough to hide phone distractions from the user. While he could not leave the app, he found himself checking notifications, which could very easily cause distraction. For example, he would get worrying emails and spend time thinking about how to reply to them rather than studying. Second, the app did not cultivate a study habit. The Forest app does not attempt to discourage compulsively opening and closing the phone. Our group member found himself looking at your phone constantly while the study timer was running—he was checking to see when the timer would end. Third, our group member did not think that building the forest was a big enough incentive to study; it felt isolating, and he just wanted to look at social media instead of studying.

3 KEY REQUIREMENTS OF A STUDY APP

From our group member's experience with *Forest*, we derived three requirements that a perfect study app should satisfy.

First, **the study app needs to remove all distractions created by the phone**. Often, study apps disable the internet, distracting apps, websites, etc. The user needs to have uninterrupted work time, and the more distractions that pass through while the user is studying the worse the productivity outcome will be, so phone distractions must be minimized as much as possible. **The Forest App did not prevent the user from looking at their phone**.

Second, **the study app must cultivate the study habit**. Throughout our lives, we have found out that studying is a habit like any other, and habits require a careful mindset and mental rituals. Thus, if the study app is to help the user study, it must nurture a very strong connection with the user's mind to indicate "this is the time to study" and "this is the time to stop studying, you can rest now". **The Forest App did not do enough to encourage the development of an instinctive study habit**.

Finally, **the study app must have a tight "game-like" reward mechanism, hopefully with a social component**. Studying is hard; the user needs incentive every day to keep working with the app. And we mention the social component because study app studying is lonely and asocial—it doesn't satisfy our need to connect to others which would have been satisfied if we were allowed to browse social media instead of studying. So we want to be able to give the user this social satisfaction. **The Forest App's forest does not provide adequate social gratification**.

4 SOLUTION OVERVIEW

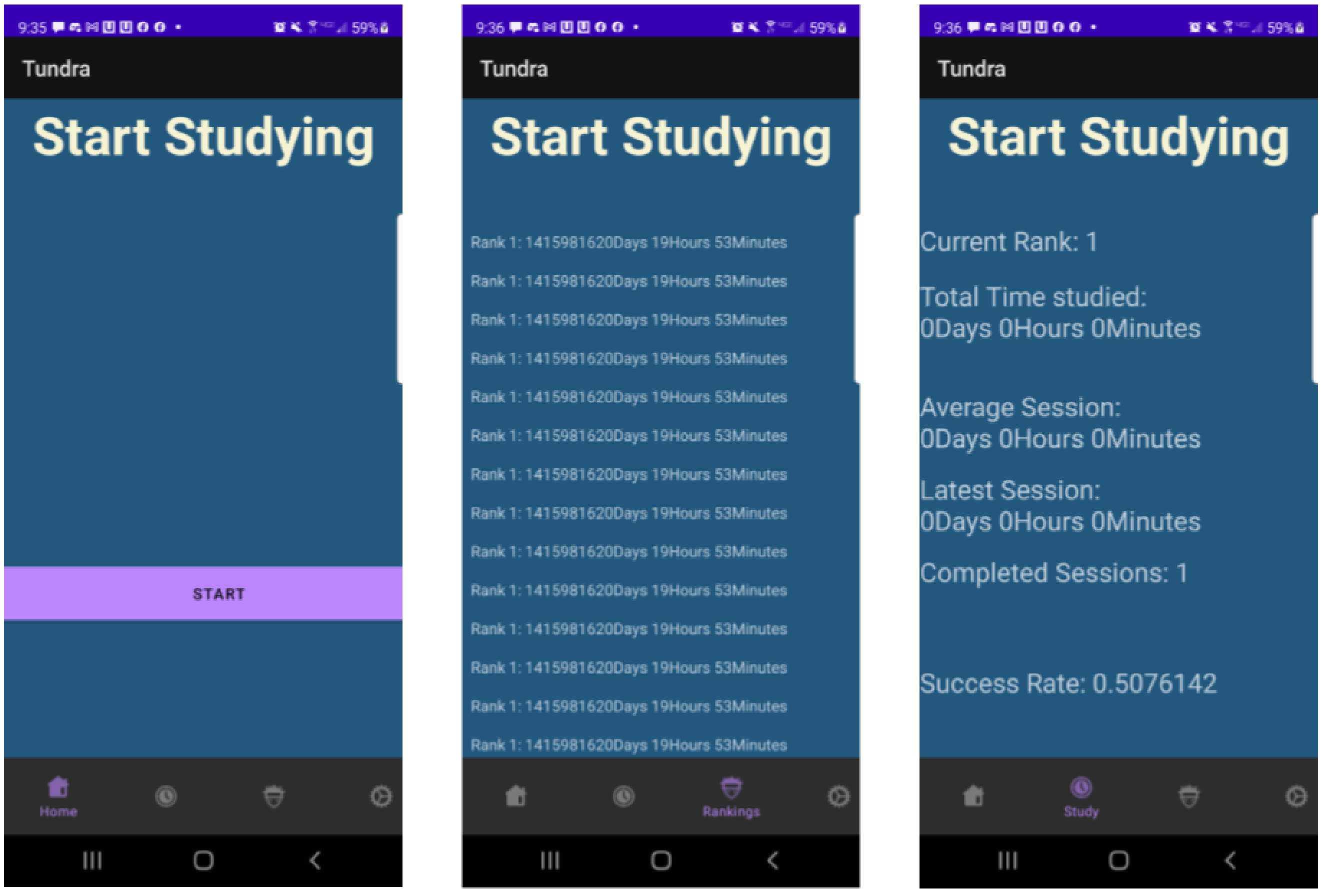
Tundra is built for Android on Java. It uses Android's fragment and activity system to pass data through the app. There are two activities—the "Study" activity and the "Main" activity. Within the "Main" activity, we used a series of fragments to create 4 distinct pages:

- Home
- Stats
- Rankings
- Settings

The "Home" page only contains a button that will switch to the "Study" activity. This is so that when users first open the app they are able to immediately jump into studying without having to traverse through pages.

"Stats" and "Ranking" are two pages we use to create the gamified social experience described previously. In the "Stats" page, users are able to see their individual study statistics such as total time, average session length, and total number of sessions. These statistics are maintained throughout the app and are updated after the user finishes a study time block.

The "Ranking" page shows their position in the "global" rankings. Rankings are determined only on total time studied.



Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes
Rank 1: 1415981620Days 19Hours 53Minutes

Current Rank: 1
Total Time studied:
0Days 0Hours 0Minutes
Average Session:
0Days 0Hours 0Minutes
Latest Session:
0Days 0Hours 0Minutes
Completed Sessions: 1
Success Rate: 0.5076142

Figure 1: Three Main App Pages ("Home", "Ranking", and "Stats")

There is a feedback loop: the user can study and rise in the rankings, but if they study less and less their ranking decays over time automatically and they lose their place. By comparing their ranking to their friends, users are given a social studying experience.

Lastly the "Settings" page has been left empty for now.

4.1 The "Study" Activity

Studying proceeds as follows:

(1) When the user opens the "Study" activity, they are greeted with a timer.
(2) After selecting a time to study, the user is instructed to flip the phone upside down.
(3) *Tundra* connects to and turns on the accelerometer sensors to make sure the phone is indeed flipped upside down.
(4) Then, while the timer is running, the accelerometer sensors keep monitoring the phone.
(5) If the phone is flipped back up but the timer hasn't finished yet, the timer stops prematurely, the phone lets the know that the study block was aborted and the user does not get a ranking reward.
(6) However, if the timer finishes without being flipped up prematurely, the phone also makes an alarm sound, the user is given a ranking reward, and the user can resume using the phone.

At the end of a study session, users are prompted to continue studying and continue the loop, or finish by closing the "Study" activity.

The accelerometer flipping mechanism guarantees that the user does not get distracted in any way—the user is literally not allowed to interact at all with the upside down phone. We originally sought to use the proximity sensor to track whether the phone was upside down, however we found that the accelerometer was far more ubiquitous within modern day phones. Also, one can trigger a proximity sensor by just putting something in front of the phone, while one cannot trick the accelerometer sensor as easily in this application: Tundra polls z-acceleration values; an upside down phone has a z-acceleration value of $-9.8m^2/sec$, and having the

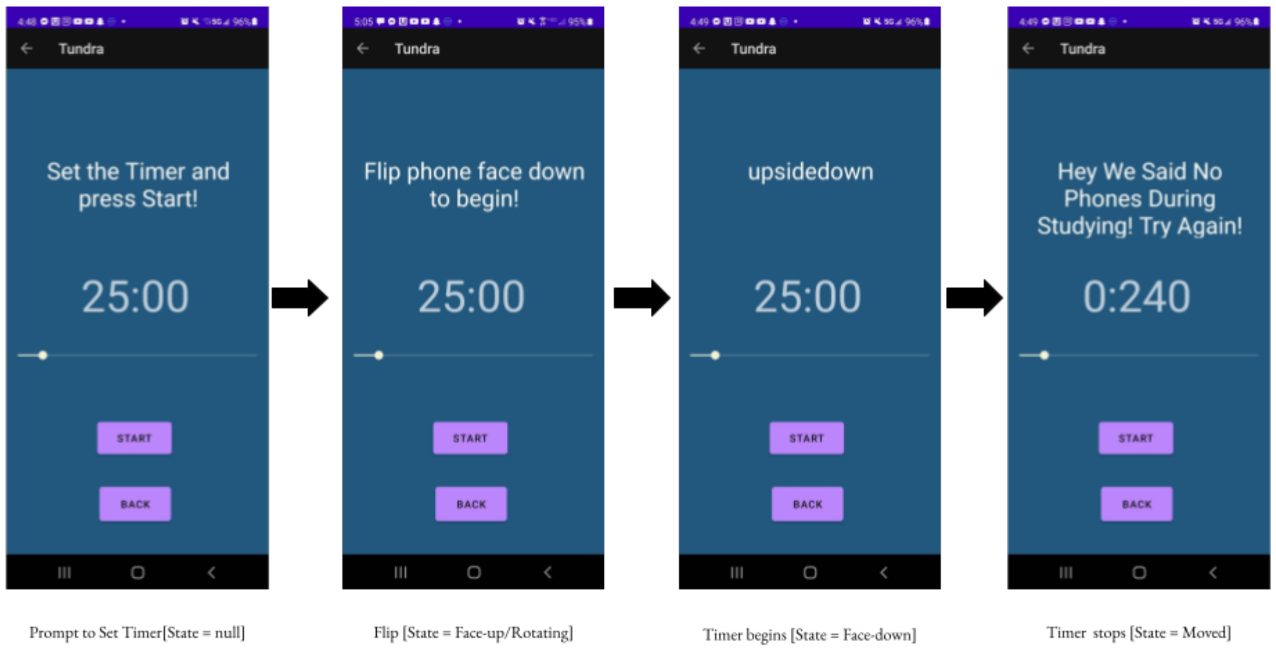

Tundra
Set the Timer and press Start!
25:00
START
BACK
Prompt to Set Timer[State = null]
Flip phone face down to begin!
Flip [State = Face-up/Rotating]
upsidedown
Timer begins [State = Face-down]
Hey We Said No Phones During Studying! Try Again!
0:240
Timer stops [State = Moved]

Figure 2: The "Study" Activity

phone in any other position will lead to a more positive z-acceleration value, unless the phone is perpetually moving vertically upwards but this is obviously impossible.

The accelerometer sensing algorithm will be studied in more detail in the next section.

4.2 An Improved Study App

As we noted previously our app has three main goals we would like to maintain throughout the design.

Firstly, we wanted to eliminate device distractions. Though the current version does not silence notifications, by having users place their devices face-down and ensuring they are not picking up their phone we are able to eliminate most visual and habitual distractions. Secondly, to cultivate clear study habits, we connect the beginning of studying with the action of flipping the device over. This helps users build a simple but effective habit to distinguish when to study and when to browse their devices. Lastly, the creation of a social game through the "Ranking" page, allows for users to gain the social satisfaction previously mentioned.

5 FLIP DETECTION

To ensure the device is face down we find accelerometer values through the commonly found accelerometer and apply a simple statistical threshold analysis.

5.1 The Z-Axis accelerometer values

Smartphones come with 3-axis accelerometers, but for our purposes we primarily focus on the z-accelerometer values that come perpendicular to the phone's screen. We set threshold values found through experimentation to see whether the phone is face up or face down.

Because the z-axis is perpendicular to the screen, if the phone is not moving and is lying either the right way up or upside down, the phone's z-accelerometer feels the standard gravitational acceleration value of $9.8m^2/s$. We can determine the orientation of the phone by looking at the sign of this z-accelerometer value. If the phone is face-up the acceleration due to gravity will cause the z-accelerometer value to be $9.8m^2/s$, however when the phone is face-down, we will see a z-accelerometer value of $-9.8m^2/s$.

5.2 States and classification algorithm

Our application actually categorizes the phone into one of five possible states, not just right way up and upside down. The states are (1) Face up, (2) Rotating, (3) Upside down, (4) Set down, and (5) Moved and interrupted. Each one can move to the next state if some threshold conditions are met. Here is the process in general:

- Start with State=Face up.
- If State=Face up and $z \leq 0$, set State=Rotating.
- If State=Rotating and $z \leq -9.8$, set State=Upside down.

- If State=Upside down, $|x| \geq 3.0$, $|y| \geq 3.0$, and $z \geq -9.8m$, set State=Set down.
- If State=Set down, $|x| \geq 0.3$, $|y| \geq 0.3$, and $z \geq -9.8m$, set State=Moved and interrupted.

5.3 Walkthrough of example flip

A user takes their phone and begins to flip it. Once it is past the halfway rotation point, the state switches into Rotating since $z \leq 0$ since the phone points slightly downwards. Actually, this rotation state has little algorithmic value (since the next check is $z \leq -9, 8$) except that the UI changes to tell the user that it detects flipping. Next, the user keeps turning. We expect the phone to be slightly falling down if the user is tilting downwards in a standard flipping motion. Therefore, during this fall the z-accelerometer value shouldn't go below -9.8.

When the phone falls on a surface, the z-accelerometer value dips momentarily below -9.8 since the phone stops but the z-accelerometer wants to keep falling. Therefore, the phone enters the Upside down state. Now, we give the user a bit of leeway: the user is allowed to move the phone significantly (in ANY of the three directions) so that $z \geq -9.8$ or $|x| \geq 3.0$ or $|y| \geq 3.0$. If the user does this, the phone enters Set down state and the accelerometer thresholds become much stricter—the user is then not allowed to move the phone at all.

If the user moves the phone aggressively again, the phone enters state Moved and Interrupted. The alarm sounds and the study block is aborted.

Accelerometer readings are not continuous—in our testing they appear every $200ms$. We were worried that if the accelerometer readings were big enough to make a transition to the Set down state for example, then the next accelerometer reading might still be huge and trigger the Moved and interrupted state accidentially even though the user moved the phone once and was going to stop. However, we have for this purpose several intermediate steps–Rotating, Upside down, and Set down, and if accelerometer readings come every $200ms$ the user has $600ms$ (from the start of Rotating to the end of Face Down) to stop moving phone, which should be plenty of time.

This procedure seems to work. In the next section, we evaluate the procedure using our phones to make sure we get expected behavior in real-life scenarios.

6 EVALUATION

The main advancement of the Tundra study app is the flipping sensing. Flipping the phone allows easy activation of the timer, and predictably puts the phone upside down. Then, once it is upside down, the phone's accelerometer sensors keep running, making sure that the phone is not flipped by the user the right way up. We have three main testing questions:

- If the user flips the phone upside down, does the timer start (in other words, last state is Upside down or Set down)?
- If the user does not flip the phone back up, does the timer keep going (in other words, last state is Upside down or Set down)?
- If the user does flip the phone back up and the timer hasn't finished, does the timer stop (in other words, last state is Moved and interrupted)?

We're going to consider these three testing questions in the following six scenarios:

- The user flips the phone normally.
- The user flips the phone onto a slanted 11° surface.
- The user flips the phone onto a slanted 35° surface.
- The user flips the phone very slowly.
- The user flips the phone very quickly.
- The user accidentally shakes the table while the phone is upside down and the timer is running.
- The user does not flip the phone up but instead raises it very slowly to look at it from underneath.

Figure 3 and 4 show sample accelerometer readings of these five scenarios, tested on the Oneplus 7T Pro. In each experiment, we flipped the phone upside down and flip is back up at the end. **Note that the procedure may not go through all five states (if the timer is not aborted) or even through the first four states**. If the user does not move the phone aggressively at all the phone will remain in the Upside down state for example.

6.1 Discussion

From the sample graphs we provide the following discussion:

- Normal flipping just works.
- Slanted flipping does not work. In the case of 11° degrees, the y-accelerometer value is huge because of the slant. In the case of 34° degrees, the phone never entered the Upside down state because the big slant did not allow the z-accelerometer values to reach -9.8. This is not a problem for us; the user should just use a level surface.
- We encountered no problems with flipping quickly. A quickly flipped phone is still upside down. Flipping it back up quickly creates huge accelerometer values across all x, y, and z.
- We encountered no problems with flipping slowly either. The phone remains in the Upside down state since it isn't moved aggressively. When the phone begins to flip and the z-accelerometer value climbs, the timer stops as expected.

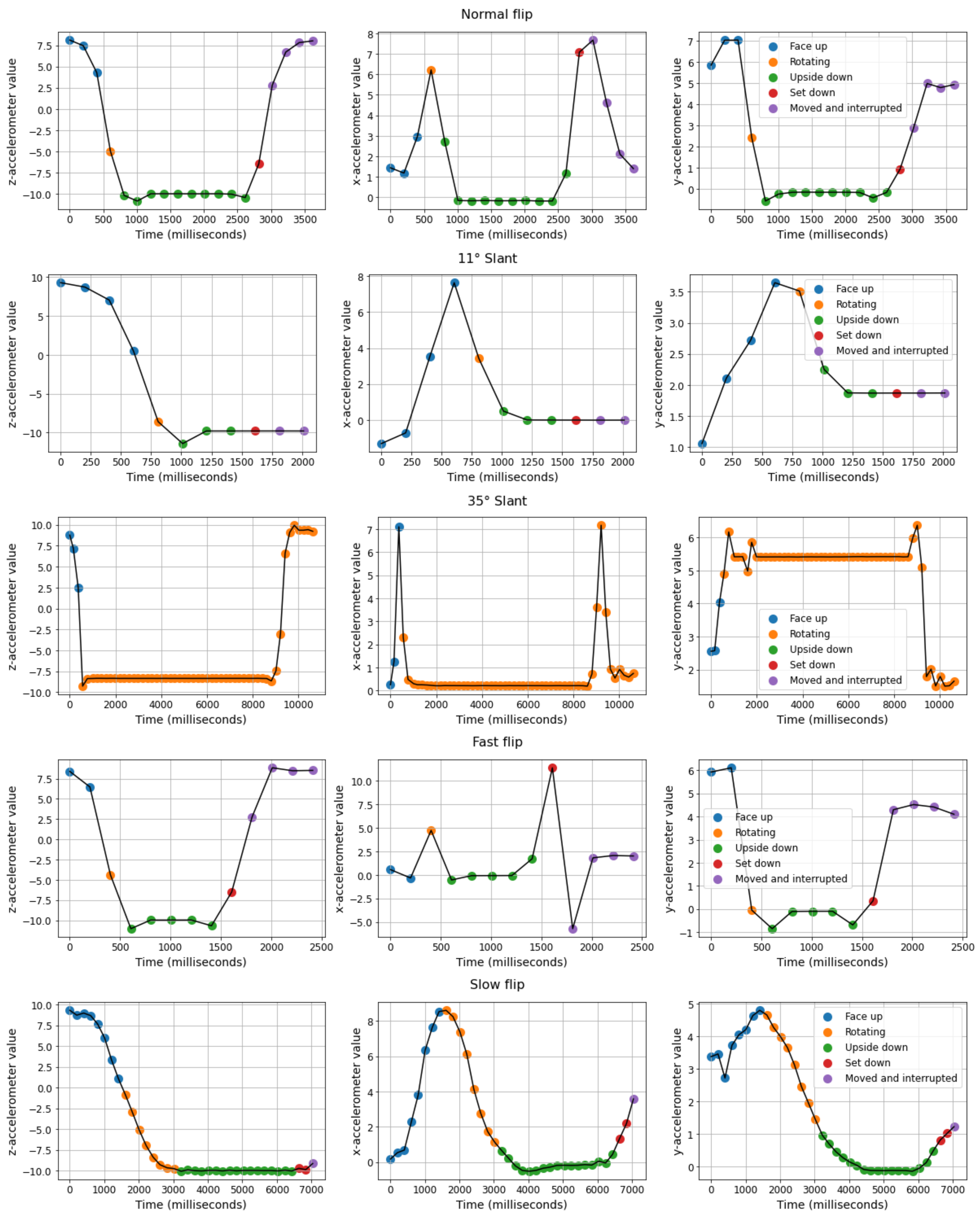


Figure 3: Sample Accelerometer Graphs

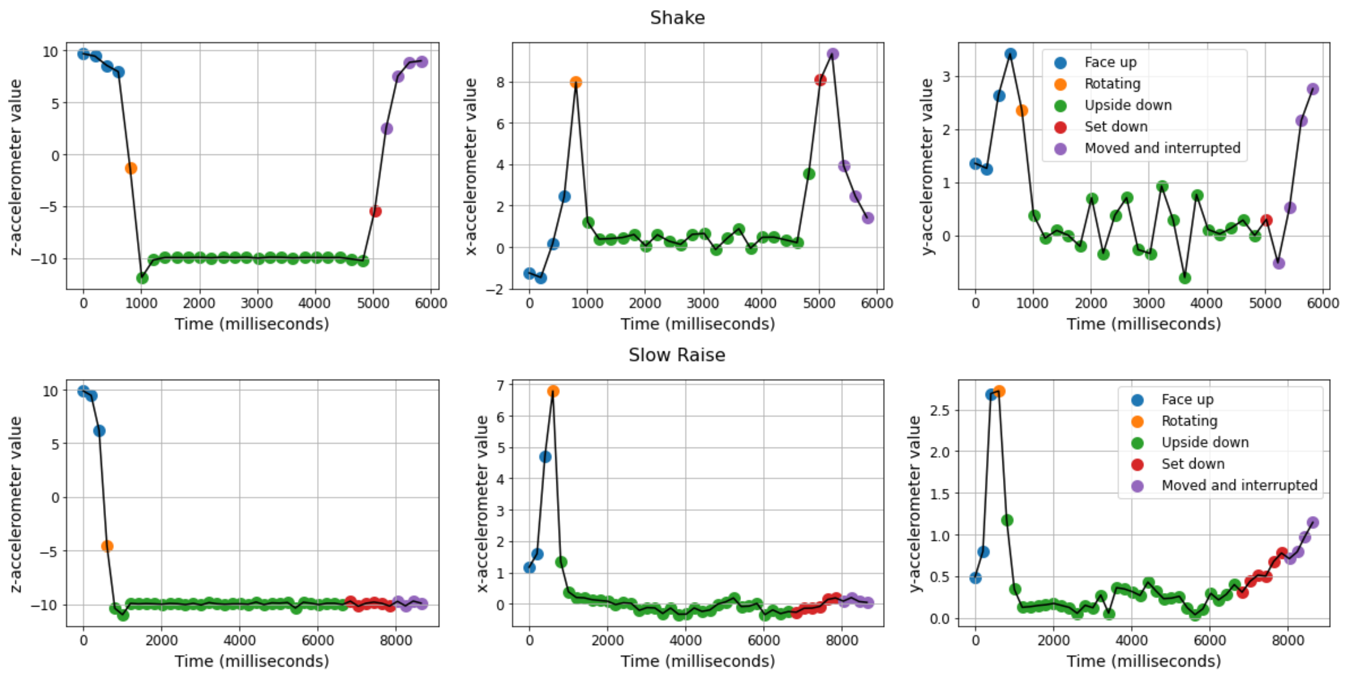


Figure 4: Remaining Sample Accelerometer Graphs

- We encountered no problems with shaking either. The threshold value of 3.0 for x and y are large enough not to trigger the Sit down state.
- Slow raising might be a way to cheat the system. However, even though I was careful I still triggered the -9.8 z-accelerometer value threshold, which stopped the timer.

In conclusion, across every single edge scenario that we have come up with, the sensing works as expected for all three testing questions. While the flipping mechanism can be potentially tricked with the right setup, we thought of the most important edge scenarios and in every one it meets our expectations.

It would also be helpful to know that the flipping mechanism is consistent across normal scenarios. We tested normal flipping, fast flipping, and slow flipping, 10 times for each scenario. An experiment contributes positively to accuracy when flipping down leads to Upside down or Set down state, not moving does not change the state, and flipping up leads to Moved and interrupted state. The results are in Table 1. We are happy to see the flipping mechanism is very consistent.

7 FUTURE WORK

Tundra needs polish in UI; the ranking system doesn't connect to a server; the app does not scale well across devices with different screen sizes; ranking does not reward the user in a game-like way; the timer stops if the screen is locked; *Tundra* cannot silence notifications yet. These are all very important missing features that must be completed before releasing the app for the public.

Table 1: Accuracy of scenarios with 10 realizations each

Scenario	Accuracy
Normal Flip	100%
Fast Flip	100%
Slow Flip	100%

However, here are three more scientifically interesting areas of work that would lead to key features of the perfect *Tundra* study app.

First, it would be amazing if Tundra could be put in your pocket instead of forced to lie on a table. Some people want to work while walking around. For this purpose, we propose to use LIMU-BERT[4] to classify whether the phone is in the pocket (not distracting the user) or not. LIMU-BERT is the BERT model specifically adapted to accelerometer data; it is energy-efficient and sophisticated, and it can really work for our use purpose.

Second, it would be amazing if *Tundra* were battery-efficient. We need to poll the accelerometer sensor data often but maybe there is a workaround. We could use EPROF[2] to build a fine-energy account of Tundra's energy usage and test how different polling leads decreases or increases energy drain. From the article we know that most energy is taken up by IO operations such as sensor reading; maybe we can rebuild *Tundra* using a lower-level library to optimize and minimize polling operations.

Finally, it would be amazing if the user can extend the timer by 5 minutes, or check how much time is left by touching the back of the phone. Such fine gesture control is possible with Acoustical Structure-borne Propagation[3]. A flipped down phone is in prime position for this gesture-based control, and if this is control were consistent and easy to use the app would be really impressive.

8 MEMBER CONTRIBUTION

- Marc Maliar—created Latex Template, ran experiments, generated accelerometer graphs, wrote Abstract, Section 1, Section 2, Section 3, Section 6, Section 7, and 70% of Section 4 and Section 5.
- JR Ansera—Developed the Back-end, as well as the Initial Front-end of Tundra, also developed the initial threshold values used in our sensor data. Created app mock-ups and graphics seen in the report as well as helped write initial draft of Sections 4 and 5.
- Francis Yang—Basic timer function and UI, fine tuned front end components (including recycler view for ranking and display of statistics on study section).

ACKNOWLEDGMENTS

To Professor Zheng, for a great mobile computing class.

REFERENCES

[1]]Forest [n. d.]. Forest Website. https://www.forestapp.cc/. Accessed: 2022-03-18.

[2] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. 2012. Where is the Energy Spent inside My App? Fine Grained Energy Accounting on Smartphones with Eprof. In *Proceedings of the 7th ACM European Conference on Computer Systems* (Bern, Switzerland) *(EuroSys '12)*. Association for Computing Machinery, New York, NY, USA, 29–42. https://doi.org/10.1145/2168836.2168841

[3] Lei Wang, Xiang Zhang, Yuanshuang Jiang, Yong Zhang, Chenren Xu, Ruiyang Gao, and Daqing Zhang. 2021. Watching Your Phone's Back: Gesture Recognition by Sensing Acoustical Structure-Borne Propagation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 2, Article 82 (jun 2021), 26 pages. https://doi.org/10.1145/3463522

[4] Huatao Xu, Pengfei Zhou, Rui Tan, Mo Li, and Guobin Shen. 2021. LIMU-BERT: Unleashing the Potential of Unlabeled Data for IMU Sensing Applications. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems* (Coimbra, Portugal) *(SenSys '21)*. Association for Computing Machinery, New York, NY, USA, 220–233. https://doi.org/10.1145/3485730.3485937