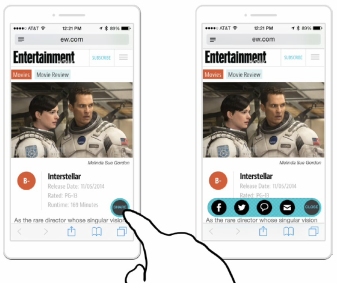
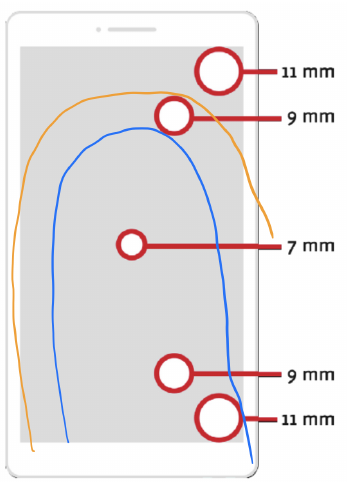
Mobile design

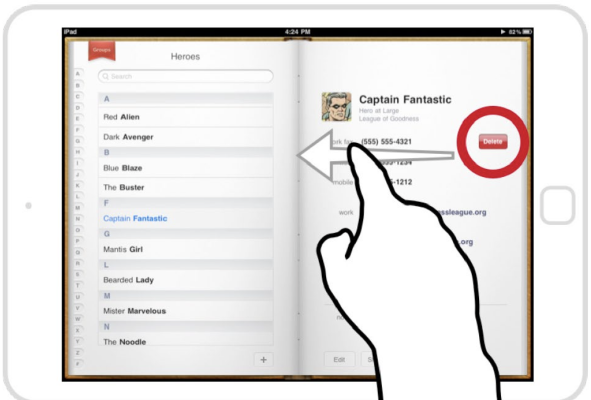
1. **Content Always on Top** rule: data must remain visible at all times, it should be not hidden by the hands
   * **exception**: where frequent data modifications may occur, in those cases the controls must be inside the comfort zone.
2. Put the most frequently used controls, in the comfort zone, or thumb zone
3. More delicate controls like **data modifications** or even **deletion** should be put outside the thumb zone.
4. **The controls** should always remain on the lower side of the screen, but interactions like swipe, or the use of floating trigger buttons should be considered for frequent operations.
   * **Exception:** on **Android controls** must be on the upper side of the screen.
5. **Floating trigger** buttons help prevent hiding other controls , making it easier to hide content and expand it only when it is needed.
6. The minimum **size** to use for a button is 7 millimeters (44px, 1% missed targets)
   * The **proximity** of the elements is crucial, since the more the elements are close, the more they have to be bigger. Also, to avoid errors, small elements need to be distant from one another. For example, buttons of 7 millimeters must be at least 2 millimeters away
7. **Just-in-time interface:** A **good interface** is the one that provides **only what is necessary at that moment**, with progressive disclosure.
   * **Problem**: number of taps needed to interact with an interface: it is important to distinguish **quality taps** and **garbage taps.** The first type are the ones that add new information or complete a task, while the second ones could be eliminated, or, at best, substituted with a gesture.
8. **Carousel** are useful only:
   * **Linear data**: like the weather app, in which the user knows what to expect.
   * **Casual browsing:** in the case of pictures or with a slideshow, they work even better if they are inserted in a context in which users know what to expect (shouldiuseacarousel.com).
   * **To break up very long forms**: in this specific case, the advance cannot be automatic.
9. **Long forms**
   * Provide the correct keyboard for a specific input.
   * Prefer a list of buttons to a menu if this one is too short.
   * Avoid too long drop down menus.
   * In case of number insertions, give the options of +/- buttons if the number is not far away from an average median.
10. **Confirmation dialogues** should be avoided because they slow down the user. **Use gestures** that are sufficiently difficult to be only intentional while remaining easy enough to be fast

## Gestures

* Always choose gesture in place of buttons if possible
  + Buttons also have the tendency to cover important content, meaning removing them could also lead to more usable space in the interface
* Is there a way to **manipulate the content directly** before using a button?
* Gestures **improve an interface accessibility** since they tolerate less precision, helping in situations in which, for example, the user cannot pay close attention to the interface or when it is needed to have a fast interaction with no errors.
* Gestures tend to become reflexes, since they build muscle memory and are not based on visual memory.
* **Gestures** are deeply linked to the concept of **metaphors**.
  + A good rule is to get help from the real world, in the sense that a gesture similar or equal to the real one has no need to be taught to the user.
  + everything must follow conventions: meaning it is extremely important to not betray users expectations.
* **Shortcuts**: complex gestures that can be used like keyboard shortcuts. (think of longer gestures, and it is also possible to use multiple fingers)
  + It is not always good, since it is sometimes not supported by devices and it is not easy to learn.
* **OS interference:** gestures by the OS cannot be used by the application. In Android the OS gestures always start from the sides, but iOS is more complex

### 

### Teaching gestures

* **just-in-time education:** we do not send the information from the start, but we wait until it is needed.
* **skeuomorphic design:** representing elements of our application with real-world elements. Better to be used **together** with **just-in-time education.**
  + Although skeuomorphic design isn’t always suited for teaching gestures, the main concept that is important to remember is the one of the metaphor: **we need to find the right metaphor to represent our application and not betray it**, in order to not confuse the users.
  + making interfaces way too realistic can be dangerous, since the user expects a simple concept: looks like → acts like, and if this line of reasoning is broken, the user will be confused. (see picture).
  + we see less and less skeuomorphic designs. Excessive realism can have the effect of limiting the possibilities of an interface.

It is a good thing to provide instructions to the user regarding the gestures. Here the different techniques:

* **Coaching**:
  + easy instructions are provided when a problem shows up for the first time.
  + Ask to repeat a gesture from 3 to a maximum of 5 times,
  + Show animations when the user stops during the interaction. **Stop showing instructions** when the user has learned the gesture
* **Leveling up**:
  + **not to teach everything from the beginning**, but to rather provide a small-steps knowledge.
  + teaches only basic interactions at the beginning and lets the users use complex gestures if they autonomously find them.
  + app must be organized in different levels of complexity
* **Power-up:** 
  + If a user does a gesture frequently it is useful to provide it a shortcut,
  + **example**: once a user has tapped 10 times on a button to access a functionality, we can tell that there is a faster gesture to access it, providing an overall sense of satisfaction.

## Other interactions with interfaces

We can nowadays interact with many sensor in our devices, not just the touch screens:

* **GPS:** to provide the user’s location for applications that include a support for mapping
* **Accelerometer, compass, and gyroscope** can be used to recognize movements of the user.
* **Sensors** like the **luminosity** one can be used for accessibility purposes.
* The **camera** can be used to collect data inside applications for translation, QR code reader, or, again, accessibility.
* **Fingerprints** reader.

## Emotional design

* **Functional**: if the user is able to complete the assigned task.
* **Reliable**: if the system works, in this case every kind of failure is unwelcome.
* **Usable**: if it is easy for the user to learn how to use the system and its functionalities.
* **Pleasurable**: if the user experience is pleasant.

Studies have shown that people are attracted to elements positioned in the **golden ratio.**

* Choosing the best size for the elements in an application may cause a reaction which is not necessarily conscious on the user's side, but is judged in a good way.

**Humanization:** something really similar to a human, is typically liked by users.

* Examples:
  + Volkswagen Maggiolino
  + MM’s, designing candies like humans improved the sails of the product ( using really big faces is a factor in the strategy of marketing: new born babies tend to have bigger heads than adults in proportion to the rest of the body, which is an unconscious information we own as human beings)

**Mascot**: Another way we can help users sympathize with our products is through the use of personalities

* It should speak, answer and act accordingly to how it is designed
  + **example**: making the mascot draw from a random pool of audio messages in a database and execute them while the user is interacting with the application, creating an engagement.
    - Some customers may understand this trick though, making in reality pretty hard to apply mascots to applications

Before creating a mascot we should create a user archetype:

* We have to **collect information about the brand image** that we want to show, which also includes the **personality**, and the **language** used by it.
* How can the **mascot** get the user attention, which are the emotions users are attracted to.

Emotions are powerful, hence need to be used accordingly.

* This last topic is pretty significant, since it deeply changes how a mascot interacts with the users, influencing the success of the application: emotions are powerful, hence need to be used accordingly
* There are no strict rules, but powerful tools include surprise, pleasure, preview, exclusivity and rewards. The best example of what preview is like is a movie trailer, which help create interest in the user.

A good example of exploiting users' emotion is done by Nike, which allows the selling of a particular set of sneakers only to subscribed members, or starts the selling at a certain time, creating exclusivity and preview.

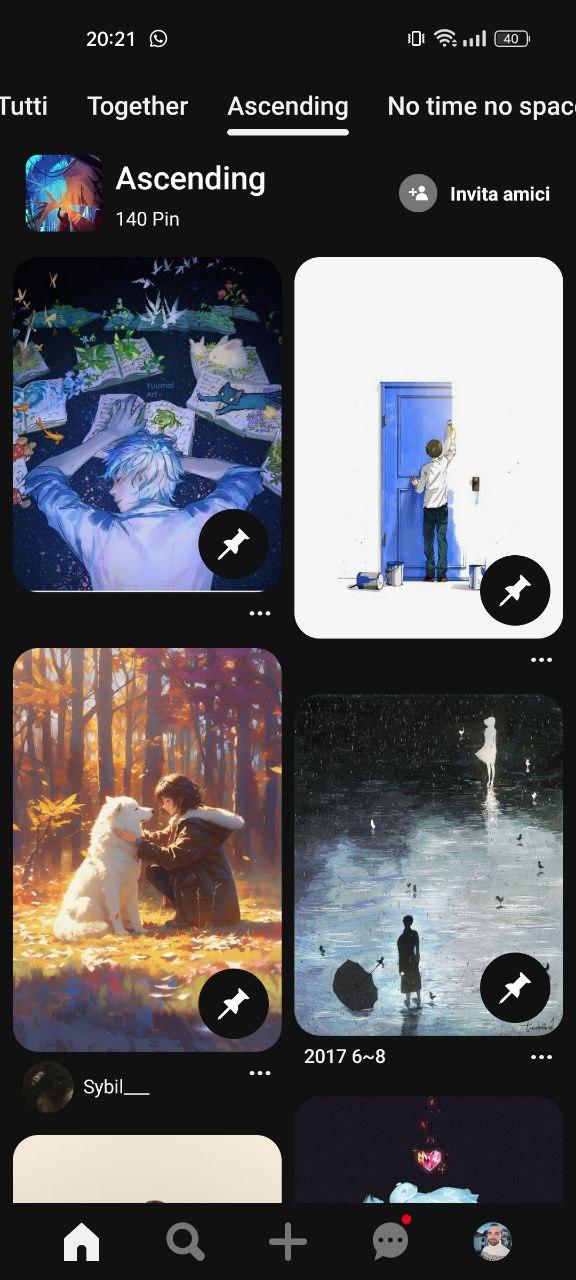
This is, though, a risky strategy, since a person will decide to consider pros and cons, but, when these are not possible to measure, will just go out on instinct, and the major obstacles for this sensation are laziness and skepticism. A correct design will help mitigate these problems, but also the use of games (think of Duolingo) and incentives can help the user prefer our option.

For all of these strategies to work we need to check multiple things about the emotions we are using.

* **Is the persona created for the brand correct**? While using a bank application we do not want to have mascots surprised at how poor we are.
* Is our product **too similar to other competitors**? This was the problem of Google+, which was way too similar to Facebook.
* **Are user needs satisfied**? For example, in social networks, if we are able to convince multiple users to stay in my application (or move to mine), since the aim of social networks is to connect with friends and celebrities.
* **Is the language correct**? It is not always possible to use informal language in most applications, but generally when we have a disruption we should tell the truth directly (the 404 error).
* **Is my application still usable, enjoyable and reliable**? After trying to invest into emotions to make the application more usable, are we still respecting all the other, fundamental, services?

# Applications

## Pinterest

* Content is always visible and in the okayish/accurate zone
* Controls like adding and modifications are easily seen and reachable when needed, in a menu on top, respecting Android rule to be on top of the screen
* Interface is pretty much clean, having only images, clear pins on each image to select them
* Skeuomorphism here works given the social concepts: pinning - saving posts a user likes - simple and easy in a good way
* No need for leveling up or coaching, since UI provides fairly easily controls to make the user happy according to what he finds

### Gestures

* Long tapping is supported, to allow for other edit controls and a context menu
* Swiping left and right between different images or different tabs is present
* Scrolling up and down is supported
* Pinching in and out of images is supported
* Pull-to-refresh is present
* Going back from main content can be done both with UI and OS gestures

### Usability

* Navigation in tabs helps in content separation and navigation
* Content in grids is spread vertically, encouraging vertical scrolling
* Carousels are not present
* Lazy loading is present having images loading incrementally with some animations
* There is no need for floating buttons, since one pretty much most of the time uses the app for browsing and not mainly for creating content; interface is already as clean as it is, offering a just-in-time experience
* On the con: some users might get lost the first time they click on an image and opens sometimes the browser with the source, other times finds similar images; might require some other metaphor telling him that

### Other things

* No humanization, given the persona is formed from algorithm usage of the app
* The logo is a “brand” one
* Metaphors are pretty much understandable

## Google Calendar

## 

### Gestures

* Long tapping/Pinch zoom in and out/Rotate is not supported
* Swiping between months is possible, in days view it’s possible to scroll up and down
* Double tapping allows for days

### Usability

* Content is not always on top, given it’s presented in “interesting” ways: the default view is by month, then by randomly tapping on a day, you see it expands to a confusing view, which requires you to use the left hamburger menu and go back from the action; if the user licks on OS gestures goes away and no back button is present
* Content is clickable and the safe zone is mainly showing an event, but space is not properly used
* Days and other stuff might be hidden and other controls by the keyboard by default when opening a specific event of some kind, making the user lose focus from what he was doing
* There is a floating button useful for creating events or to create other events which the user might get from the visual metaphor of the ion but are not clear unless the user tries himself
* To delete or handle events differently, there is the pen on top and the settings option available accordingly
* The UI itself is pretty confusing, not giving any clue of where to go
* Consider implementing more common gestures like long-press for additional options, pinch-to-zoom for adjusting view scale, and rotation support for different orientations
* Utilize a navigation drawer or bottom navigation bar to surface key sections (month, week, day views) while keeping content visible

### Other things

* No mascot to report here
* There is some kind of persona formed, humanizing the app when creating events and reporting national holidays, happy birthday and some events in the coherent Material UI design by Google
* The logo is a skeuomorphism of the app itself, simple and nice
* It’s not that clear to a normal user how to handle single events, tapping on them and relying on intuition when switching from one view to the other, leading to other confusion
* Coloring of events is pretty hidden and should be clearer
* Falling short on presentation does not help browsing or presenting with carousels
* There should be some kind of coaching and leveling up technique to guide a user when implementing a different UI design
* Too much cluttered, too less space

## 

## Google Drive

## 

### 

### Gestures

* Tapping for selecting normal files
* Press and drag available to select a file and delete it
* No rotation/presses/flickering available
* Pinch-to-refresh available
* Scrolling down and up to reach new files and folders
* Swiping left and right to reach different tabs accordingly
* No need for shortcuts, since no complex gestures are employed

### Usability

* Floating trigger available describing precisely actions: uploading files and folders (should be written better the uploading file option - but conveyed by visual metaphor in some way) or usage of Google Suite apps
* On the just-in-time part, the app is not bad really, but for a normal user might be confusing having only a list of files and the expected “Repository” folder metaphor is betrayed, given it does not change anything from the main view available when the user clicks on it first time, so it might help having some kind of assistance to see things more clearly
* Content is always on top and elements are all of the same size, so proximity is not a problem
* Garbage taps can be made when selecting a file instead of one another, but the app revolves heavily around other Google apps to have back button controls or some kind of external app the user know already, which may be a wrong assumption in many cases
* Carousel is pretty linear, needed for files inside a directory but definitely not for default view

### Other things

* Other interfaces and interactions are available easier thanks to external camera access on top
* There is no example of humanization or mascot here
* Coaching needs not to be done, simply access files:only contents presentation and complexity needs to be changed here
* Edit options are always on top like the Android rule says
* The logo is a standard Google one, once again, a corporate one

## Amazon

## 

## 

### Gestures

* Zoom present on products but user may find it randomly, since it’s not clear this can be done
* Sliding between tabs is not possible
* Only possible to slide up and down
* Pressing and tapping is present to reach main content

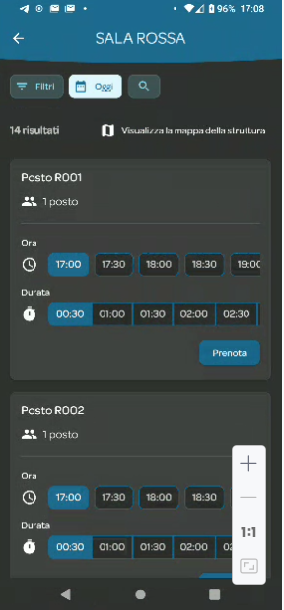
### Usability

* The layout in grid is good but fairly cluttered and quite confusing overall in its conception
* It just forces you to go down and down to see different details of products
* Call to actions are just not present as soon as you click on a product
* It expects you completely to find things out on yout own
* Back button is now so comfortable to reach and understanding to handle products and safely put them in cart or editing them requires the user to navigate multiple menus, not easily reachable by gestures
* Proximal elements are different and presented each time differently according to the context, so many colors to distract the user from the main goal
* It’s fairly easy to navigate between different ways to look at products and just buy them, with no safe way to escape from main content apart from going on top
* Content is always on top, but on this app a bit too much, given the app itself does a bad job at presenting content suitable for a mobile layout
* To have a real just-in-time interface, it should be minimized much more, breaking up the process of forms and confirmation dialogues for non-expert users

### Other things

* A small bit of humanization is present on the logo smiling, then welcoming the user to his profile section when present
* Carousels are present to browse between different images of product
* Interface is a clear representation of a product, but no skeuomorphism is made to improve usability, so one has to lose time browsing, then checking: simple but an be done better overall
* No usage of particular sensor or algorithmics to convey emotions
* No particular mascot is chosen here

## Affluences



* Back arrow hard to reach. No gesture for it but you can use the OS gesture to go back.
* Control on top but not very easy to reach. On the bottom it would have been better.
* Buttons are big enough but no separation between them, so it is needed to aim well. Overall it is easy to tap on the wanted button
* The only gesture supported is scrolling down, which can also be useful to bring the buttons near the comfort zone
* You could accidentally tap on buttons as there are many of them spread everywhere

## NewPipe

### Gestures

1. For full screen you can turn your phone. This behavior can be changed form the settings
2. The control button below can also be accessed by swiping left or right
3. When the video is playing in full screen you can adjust the volume and the brightness by swiping up and down on the left side of the screen, for the former, and the righter side of the screen for the latter. There is no training though, so if you accidentally change the brightness or the volume by mistake you are lost and you do not know what happened to your phone. The only way to learn about these gestures is to go to the settings where you can also change or disable them.
4. By swiping right from the leftmost part of the screen you access the application menu
5. Some of the controls are in the middle of the screen in the comfort zone. The content you cover are the comments, but if you are accessing those controls you are not interested in the comments, and there is nothing that can happen if you accidentally tap on them.
6. To go back to the home screen you can scroll down from the video on the top part of the screen to put it away on the background if it is playing.You can also rely on the OS gestures to go back that will also put the video on background
7. Double tapping on the video on the rightmost or leftmost part will fast forward or backward the video by 10 seconds. If you tap repeatedly it also increases by minutes.
8. Long tap has different behaviors according to the element you long tap on: On comments it copies them notifying the user; on icons it shows the label, on videos on a feed it shows a context menu with different possible actions.

### Usability

The search button is out of the comfort zone

I stop the analysis here as it is not perfect because of the lack of training for gestures. If we do not find a perfect one then I can continue with this one

## Fitotrack

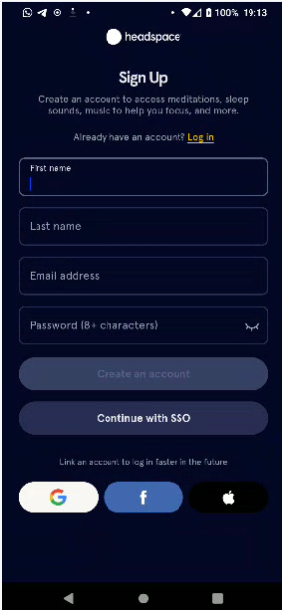
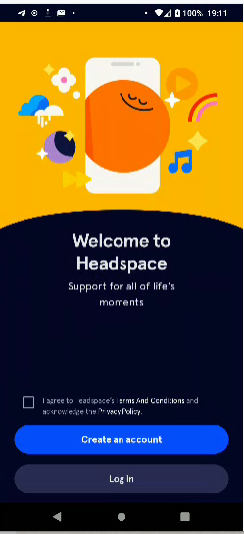
#### Gestures

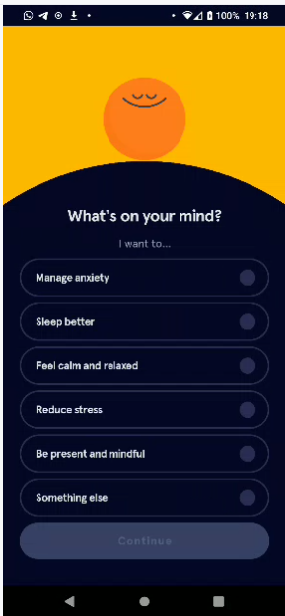
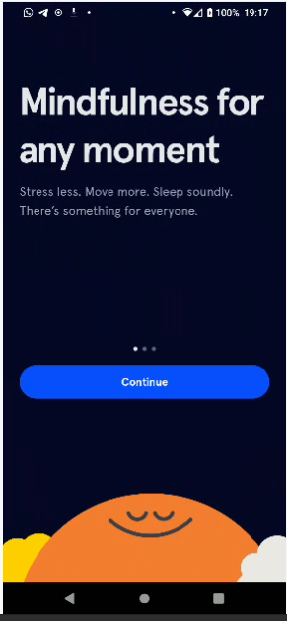
1. Long tap on a track triggers a delete dialog. Long tap is still a complex enough gesture but as the app is a list of tracks it could be easy to long press by mistake on one. Probably it is not the best gesture for it. As it is a dangerous operation it should be out of the comfort zone but the dialog box, although it slows down the user, protects it from accidental deletion.
2. The other gesture is scrolling up and down
3. Going to the previous screen provides no gestures, it relies on the OS one. But there is a back arrow on the top left corner which is difficult to reach
4. Scrolling right and left to move through the tabs on the top

#### Other important things

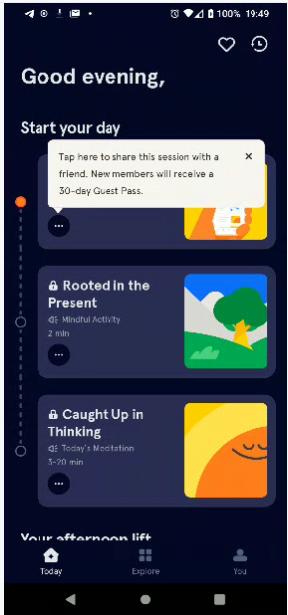
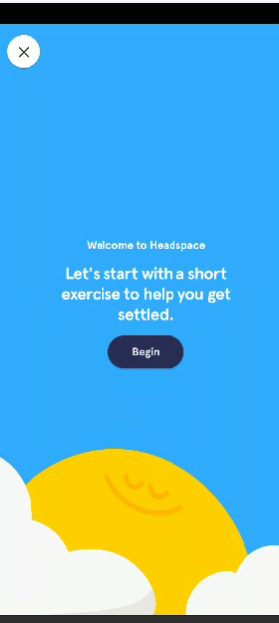
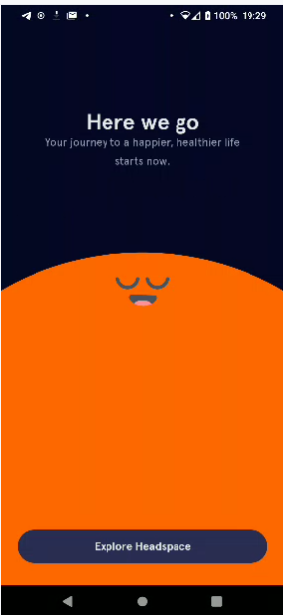
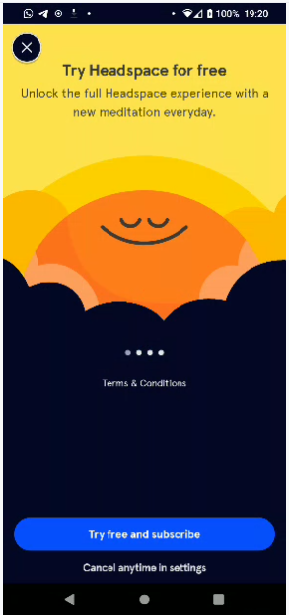
1. Settings (three dots on the topmost right) are out of the comfort zone and difficult to find. As settings are not an everyday operation it is not a big deal. It could be improved by adding a scroll right gesture.
2. It has a floating trigger on the bottom right with metaphors
3. The metaphors are not all appropriate but they are combined with a label. The + symbol for adding a new trek (good), The pencil to insert manually a trek which by convention can be confused for “edit” a track. The gray plus sign for importing a new track is also confusing. And finally the cross symbol for collapsing the floating trigger is ok.
4. No humanization in the app
5. The logo is a appropriate a it describes very well the use it is for
6. No persona

## HeadSpace





# 



# 

# Assignment

Create groups of two students and create a presentation with 2/3 slides with:

* an example of positive design of a mobile application (correct usage of gestures, proper metaphor, the use of correct instructions, just-in-time content, etc.) and
* an example of a wrong design.

Provide at least one screenshot for each of the two examples and a brief description of the main points and why the design is correct or incorrect. For the wrong example, provide a possible solution to resolve the problem.

The present