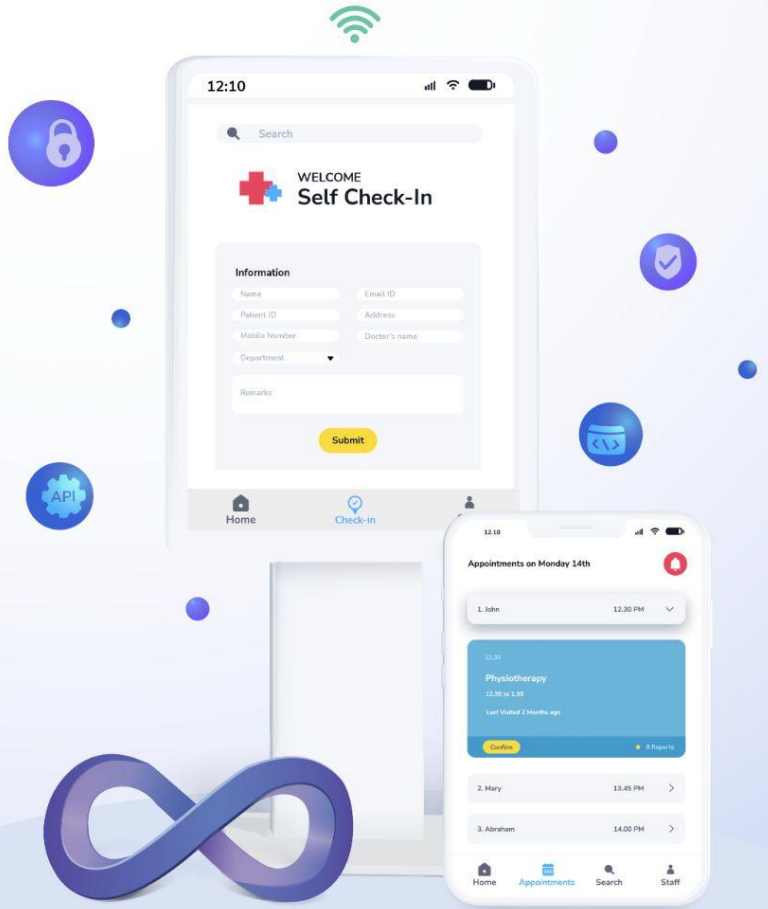
esper

Esper Interactive Guide

Choose the best OS for your dedicated HealthTech devices

Get Started



Choose your path

Not every OS is up to the task of managing dedicated hardware, especially in the healthcare space where devices need to be always on, always secure, and always up to date. In this guide we'll take you through the highest impact considerations across four main OSs that will help you find which one is right for your fleet of dedicated HealthTech devices.

See direct comparisons

- OS overview
- OS comparison

Dive into a concern area

- Design
- Technology
- Cost
- Resources
- End-user experience

Learn about a specific OS

- iOS
- Linux
- Windows
- Android

See direct comparisons

OS overview

Click on an OS to see a full overview of its characteristics or click ahead for an in-depth side-by-side comparison.

iOS

Premium hardware, but rigid control

Expensive hardware that delivers premium experiences. Mostly targeted at the prosumer market.

Apple offers rigid constraints that you'll be required to work within.

Linux

High flexibility with high effort

Heavily focused on IoT with very few finished devices available.

Affords very high control and customization, but you'll take on the overhead and costs of creating your hardware.

Windows

Slow moving, legacy technology

Longstanding OS that wasn't built for modern healthcare use cases or dedicated devices.

Driven by PC world and primarily focused on x86. Low innovation and experimentation outside of this.

Android

Modern and flexible, but quality varies

Wide range of options with flexibility to tailor to your use case. Has a broad and highly active ecosystem.

Quality can be variable, especially in low range options. You'll need to vet the hardware you're selecting.

OS comparison

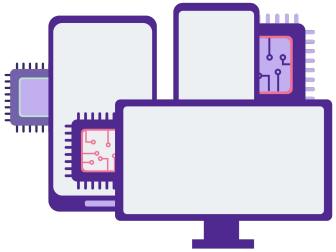
See how the 4 operating systems compare across 5 main categories of considerations. Click into a consideration category to get more details or click an OS type to see a full overview of its characteristics.

	iOS	Linux	Windows	Android
Design	Rigid model options with fixed yearly release schedule.	Complete design control with hardware available long term.	Designs based on PC form factor with few long term hardware options.	Diverse design options with hardware available long term.
Technology	Beholden to Apple's proprietary technology. Strong, proven security.	Wide open and flexible tech set, but IoT focused. Strongest kernel imaginable.	Not built for dedicated devices. Good enough security for enterprises.	Flexible tech set with fine controls. Security based on device classification.
Cost	Expensive hardware for premium experience, creating a higher overall TCO.	Highly variable due to flexibility of options. Limited finished products available.	Semi-flexible, but high floor due to x86 and Windows licensing fees.	Wide range of options. Costs driven by the hardware you choose.
Resources	Large developer pool, but limited and closed ecosystem.	Wide open ecosystem if you can find find and hire Linux developers.	Ecosystem centered on x86 on Windows. Lower developer activity.	Broad and flexible eco-system with diverse developer pool.
End user experience	Familiar UI to certain users. Uniform robustness level across hardware.	Must build own UI, but get full branding control. Robust and resilient hardware.	Not a modern UI. Performance varies for dedicated use cases.	Most popular UI. Get full branding control. Quality of OS implementation can vary.

Dive into a concern area

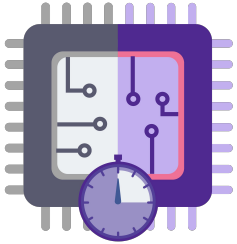
Design

Dive into the 3 main design factors to consider when choosing an OS for your dedicated devices. Click on a consideration factor to see a side-by-side comparison of the OSs.



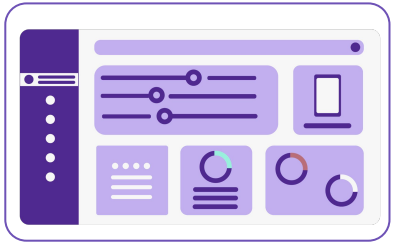
Hardware diversity

Range of hardware and design options available.



Hardware longevity

Length of time hardware models will be available.



UX design flexibility

Customization level available to create your user experience.

Design: Hardware diversity

	iOS	Linux	Windows	Android
	Only fixed options available	**Full control to create custom designs**	**PC focused with minimal experimentation**	**Wide range of options available**
Pros	Apple delivers homogeneous behavior across devices. If the fixed option available can work successfully for all your use cases, this could work for you.	Full control to build whatever you want here. Great for developing fringy or leading edge healthcare use cases.	Experimentation can deliver some interesting designs for the PC form factor.	Unparalleled diversity from a broad ecosystem. Have flexibility to choose the right design for your use case. Can access the benefits of Linux through AOSP.
Cons	Offers minimal design language or use case diversity. Differentiation here is more speeds and feeds. Not best for fleets that have a diverse set of medical devices.	Few finished devices available. You'll need to take on full ownership of the design and costs. Can have long lead times so not best for building and getting to market quickly.	Innovation driven by Microsoft, not the ecosystem as a whole. Designs are primarily for laptops, desktops, and servers, which won't give you high variety or serve telehealth use cases.	Need to be careful with hardware at the low end of the range. Be sure to vet the hardware you're selecting to ensure it will deliver the performance and security required in the healthcare space.

Design: Hardware longevity

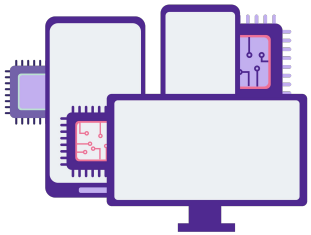
	iOS	Linux	Windows	Android
	Fixed yearly model refresh cycle	**Driven by hardware availability**	**Based on PC refresh cycle and silicon delivery**	**Short and long term options available**
Pros	A new model each year means you can deliver patient and staff premium experiences on the latest and greatest hardware.	Hardware tends to be available longer, making it easier to standardize on a particular model— beneficial for cases like clinical trials where you need to lock in specific hardware.	Although long term delivery can be hard here, a lot of vertical market devices delivered from OEMs have longer support.	Refresh with new releases or get longevity from commercially oriented options to fit use case. Build custom with AOSP to get longer term availability of silicon.
Cons	Older models not available long term so you can't standardize on a model at a specific price. This makes it complicated to build a fleet and maintain compliance and performance.	Will need to keep pace with market availability. Market is not designed for finished devices and is primarily focused on boards.	Over time you won't get the hardware needed to deliver your solution because of refreshes. Silicon delivery and new designs from Intel and AMD also a factor here.	Will need to navigate the market to find your right fit. Building custom will require negotiating availability timeframe with your supplier.

Design: UX design flexibility

	iOS	Linux	Windows	Android
	Lowest flexibility available	**Ultimate flexibility**	**Semi-flexible, but stale**	**High flexibility**
Pros	Works if you're comfortable with Apple's design constraints and long-term product roadmap.	Best option available in terms of customization. Can design a fully custom UX tailored to your use case and user needs.	Gives you some good design flexibility if you're happy with the core experience Windows provides.	Lots of tooling available to build the UX you want for the user experience, especially with AOSP. Many devs can create apps compatible with Android.
Cons	Locked into overall iOS UX which spans the UI, OS, and hardware. Will need to ensure your roadmap and Apple's (which is optimized for the prosumer market) align long term.	Customization brings more complexity and cost. You're on your own to plan and design the UX, and you'll need to pay your engineers to create it.	Wasn't designed for touch or modern use cases, like telehealth. Committing to Microsoft's path and the uncertainty of them investing in smart devices or ARM.	Not as flexible as Linux. Has high customization, but can't fully customize the UX.

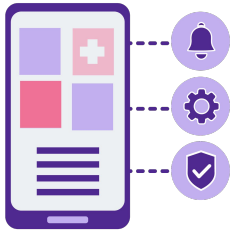
Technology

Dive into the 4 main technology factors to consider when choosing an OS for your dedicated fleet. Click on a consideration factor to see a side-by-side comparison of the OSs.



Technology set

Full overview of the entire technology set provided.



EMM Infrastructure

Full overview of the entire technology set provided.

Infrastructure for apps

Ability to manage the apps installed on your dedicated devices.

Security

Level of security provided to protect your dedicated device fleet.

Tech: Technology set

	iOS	Linux	Windows	Android
	Rigid and proprietary	Ultimate customization	Established legacy technology	Flexible and open
Pros	Proprietary technology. A good fit if you're comfortable with following Apple's technology decisions (i.e. hardware design, cabling, processors, etc.).	Longstanding OS with a dynamic open source community. Can design any device you want for your use case.	Longstanding OS. Focused on x86 with some ARM activity.	Open and pragmatic approach to tech set. Best for executing a diverse set of use cases (i.e. remote monitoring, staff tablets, handheld clinical devices). Can standardize on one OS.
Cons	Beholden to the decisions Apple makes which are based on them optimizing for their high-end prosumer market, not dedicated or high-security use cases.	Focused on IoT world. Won't benefit from the work being done by device or component makers targeting larger consumer and enterprise markets.	Not built for modern use cases. Higher price points associated with x86. Tech reaching end of the road for overall code base and where the industry is going.	Not as customizable as Linux. Will have to navigate the wide ecosystem and vet the options available to ensure performance, compliance, and security.

Tech: EMM infrastructure

	iOS	Linux	Windows	Android
	Available	Not readily available	Available, but limited	Available with full control
Pros	Has a set of Enterprise mobility management (EMM) APIs available for remote configuration of devices.	While EMM APIs don't readily exist, they can be built using Linux by MDM providers.	If you're looking to satisfy core IT scenarios, EMM APIs are available for that.	Can get EMM APIs optimized for dedicated use cases from Esper. Access granular controls for GMS and AOSP devices via a single pane of glass.
Cons	Not optimized for dedicated devices. Don't get the lower level control you would get with Android.	Requires you to take on the burden of building for yourself what is commercially available through other options.	Built primarily for IT scenarios. Not optimized for dedicated devices.	Standard EMM APIs available on GMS devices are designed and optimized for the enterprise use case.

Tech: Infrastructure for apps

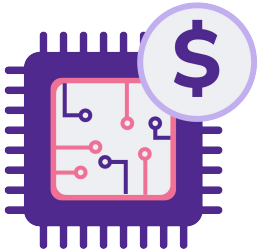
	iOS	Linux	Windows	Android
	Limited control	Highly flexible	Focused on legacy IT	Flexible and mature
Pros	Can work if you're okay with what Apple makes available. TestFlight is available to you (but it's really designed for you to test your app, not manage it).	Very flexible infrastructure. Have a good deal of control when managing apps.	Although not optimized for dedicated devices it can be used to manage apps on them.	Finer control to manage apps across your fleet. Managed Google Play available for GMS devices. AOSP gives you Linux level flexibility with more mature infrastructure.
Cons	Can't control apps outside of the app store. Have less management control- what Apple exposes to you dictates how you're able to manage your apps.	Dependent on whatever infrastructure was built for that particular Linux build from the supplier.	Focused on legacy core IT scenarios such as PC and Desktops. Not great for modern healthcare use cases that require kiosks, tablets, handheld devices, or display devices.	Dependent on whatever infrastructure was built for that particular Android build from the supplier.

Tech: Security

	iOS	Linux	Windows	Android
	Strong	Industry leading	Functional	Levels vary
Pros	Stable security with 6-7 years of patches typically provided per model (leads industry). You can fit into the ecosystem so you get releases before they come out.	Strongest kernel imaginable. Strengthened through open source community. Often used in high-security verticals such as healthcare and government.	Good enough security for the enterprise use case.	Monthly security patches based on Linux for the last 4 OS versions Google has released. Esper Foundation for Android gives you security patches for the life of your equipment and beyond.
Cons	General consumer use case focused; may not meet healthcare compliance standards. You'll get less transparency from Apple as their main concern is iOS being attacked.	You'll have to follow the community and be apart of it to participate. This means burning your engineers on this, which may not be where you want them to focus.	Will have to work within the parameters associated with keeping your devices safe through Microsoft.	Dependent on OS version you're getting so be sure to vet before buying. For AOSP, it's up to the manufacturer to incorporate security patches.

Cost

Dive into the 2 main cost areas to consider when choosing an OS for your dedicated devices.
Click on a consideration factor to see a side-by-side comparison of the OSs.



Hardware cost

Cost of the hardware associated with each OS.

Operational cost

Flexibility of operating costs associated with each OS.

Cost: Hardware costs

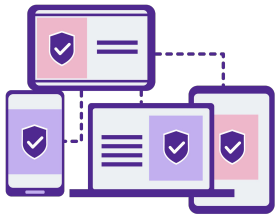
	iOS	Linux	Windows	Android
	Most expensive	Variable and complex	Tends to be expensive	Flexible
Pros	Hardware available is high quality and will deliver premium staff and patient experiences.	A wide variety of options available gives you lots of flexibility in choice. Can get lower cost computes for higher cost efficiencies.	Can find economical options, but tends to have higher floor.	Wide range from premium to really inexpensive to choose from. Mid-range options available to find your ideal performance to cost ratio.
Cons	High price point with minimal range. Won't find mid to low market options. No alternatives, you're locked into prices set by Apple based on their prosumer market goals.	Designed primarily for IoT so minimal finished devices available. On your own to navigate the market which can be complicated as it doesn't have established prices.	Tend to see higher price points with fixed form factors. x86 tends to be more expensive for silicon, plus you'll pay Windows licensing fees.	Will run into higher costs for premium hardware as these compete with Apple products. Lower cost options may not have high robustness levels needed for the healthcare space.

Cost: Operational costs

	iOS	Linux	Windows	Android
	Least flexible with highest outlay	Ultimate flexibility	Semi-flexible	Flexible with mid-range options
Pros	Can keep costs relatively consistent if you stay with one model and you're okay with Apple's yearly refresh cycle.	Lots of flexibility. Costs highly dependent on supplier relationships, Linux development resources, and robustness of your design, which you get to control.	As Windows is a long-standing OS, you're getting known and well proven operational costs.	Lots of range and variability here. Operational cost is driven by hardware choice which is up to you. Opportunity to tune hardware to your ideal operational costs.
Cons	Higher overall TCO driven by high price point of hardware and the inability to effectively repair devices often seen in the closed Apple ecosystem.	Thin market for finished devices and you won't know how the devices will perform in dedicated healthcare use cases as there isn't any data to reference.	Focused on enterprise knowledge worker use case rather than dedicated use cases. We've seen many customers move to Android and find a better TCO.	More complicated decision process since you have a wide range of options.

Resources

Dive into the 2 main resource areas to consider when choosing an OS for your dedicated devices.
Click on a consideration factor to see a side-by-side comparison of the OSs.



Ecosystem

Availability of offerings from vendors that you can harness to drive your business

Developer pool

Size, diversity, and availability of developer talent available

Resources: Ecosystem

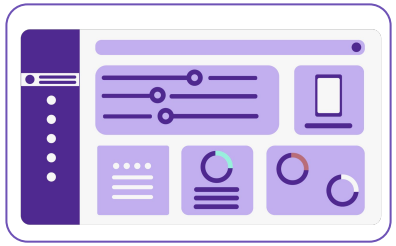
	iOS	Linux	Windows	Android
	Closed	Wide open, but IoT focused	Centered around Windows and Intel	Rich and robust
Pros	If you're happy with Apple's finished products, this could be a good choice for you.	If you have an intersection with IoT, this may work for you as the core part of the ecosystem is based off of IoT use cases.	Semi-flexible ecosystem to work with. More flexibility and choice here than Apple, as long as you align with the PC roadmap.	Broad ecosystem creating lots of flexibility. Access to core base designs driven by innovation from Google. Can apply Linux activity to Android via AOSP.
Cons	Completely closed ecosystem that Apple defines and controls. Ecosystem will be inaccessible to you.	You may have to pay more money and burn your developers time to navigate the complex ecosystem and build custom.	Locked into stagnant use case as ecosystem is driven by Intel x86 on the Windows OS.	Not as wide open as the Linux ecosystem so there's a lesser degree of customization available.

Resources: Developer pool

	iOS	Linux	Windows	Android
	Limited	Hardcore	Fading	Flexible
Pros	~3 million developers worldwide with ~2.2 million apps in the App Store to pull from.	Open source, IoT community. Good sized developer pool available to you.	Tons of Windows apps available, more than iOS and Android combined (although some may be very old).	~6 million developers worldwide with ~3 million apps to pull from for GMS. Can choose dev talent from a highly diverse set of markets.
Cons	More limited than other ecosystems like Linux and Android. Dev pool focused on what Apple is trying to build with their ecosystem.	Dependent on being able to find and hire talent, as pool is not well followed or documented. You could end up paying devs to build what is readily available through other platforms.	Not a huge amount of activity in this pool, especially when compared to the activity in the iOS and Android spaces.	For AOSP devices, you typically can't use GMS-based apps. You'll need to work with ISVs to create an AOSP version of the app or build a custom app.

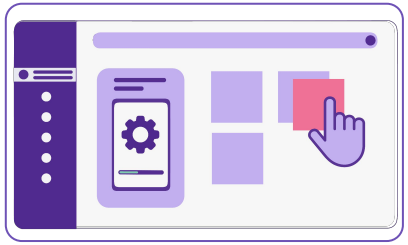
End-user experience

Dive into the 3 main end-user experience factors to consider when choosing an OS for your dedicated devices. Click on a consideration factor to see a side-by-side comparison of the OSs.



UX familiarity

Level of familiarity end users have with the UX of the OS



Brand Presence

Level of branding control and visibility

Robustness

Performance of the overall system that creates the end-user experience

User exp: UX familiarity

	iOS	Linux	Windows	Android
	Dependent on region and demographics	Low due to customization	Fading with younger users	Most known and used
Pros	Usage tied to region and demographics. Can work if the users you're primarily targeting are comfortable with it.	You have complete design control of your UI and can map it to paradigms users are more familiar with.	Some user familiarity with the Windows UI through PC devices used for work. Could work if targeting older users, as they're the most familiar with it.	Billions of users with ~70% of smart devices (phones and tablets) market share worldwide. Low barrier to learn UI if patients or staff are unfamiliar or aren't tech savvy.
Cons	Not a way to reach the global masses, especially if using tablets for telehealth or staff use cases. Only has ~30% of smart devices (phones and tablets) market share worldwide.	Not great for patient or staff facing use cases as few average users know the Linux UI. Will need to map your UI to familiar paradigms to deliver the best user experience.	Wasn't designed for touch use cases so users are becoming less familiar with it over time. Least familiar UI for younger generations.	None. This is what to use if you want an intuitive experience for most users globally.

User exp: Brand presence

	iOS	Linux	Windows	Android
	Focused on Apple	Full branding control	Flexible	Full branding control
Pros	Good for those that want to attach themselves to Apple's brand and benefit from a halo effect.	Wide open playground to create the branding experience you want to deliver.	Mid-range option with flexibility to hide some of the Windows branding from your end-users.	Equal to Linux in terms of branding. Have the ability to tweak and tune the brand experience to a diversity of device types.
Cons	Main focus is Apple's brand which will come strongly through whatever dedicated device you deliver. Brand is built into the design of the device.	Devices are completely unbranded. You are required to take on the burden of creating this instead of leveraging ecosystem resources.	At some point you will get the Blue Screen of Death and users will know it's a Windows based device.	None. With Android you get the same level of control you'd get with Linux while leveraging the broad Android ecosystem.

User exp: Robustness

	iOS	Linux	Windows	Android
	Known and stable	Heavily varied	Low for dedicated devices	Dependent on implementation
Pros	Well documented, uniform delivery of robustness. Overall system has relatively stable behavior due to having many users.	Usually see a high level of resilience as this market is IoT driven.	Has some applications where robustness is higher, such as on the server side.	Lots of choices available. Major OEMs are usually competing with Apple and therefore driven to provide the same high level of performance.
Cons	Fixed form factor that you can't get around. Performance and durability designed for prosumer use cases, which may not stand up to healthcare industry standards.	Robustness a bit of an unknown for finished devices as there isn't a lot of available data on their performance and durability.	Windows doesn't often stand up to the reliability and stability needed when deploying devices into the healthcare space. At some point you'll get the Blue Screen of Death.	Need to test device performance level before buying. Go with well known products whose ruggedness has been tested sufficiently.

Learn about a specific OS

iOS

	Design	Technology	Cost	Resources	End user experience
Pros	If you're comfortable with Apple's design constraints, yearly hardware refreshes, and long-term product roadmap focused on the prosumer market, this could work for you.	Strong long term security with industry leading support. Works if you're okay with Apple's tech set and the limited tools they make available to you.	Hardware available is high quality and will deliver premium experiences. Can maintain relatively stable costs to optimize OpEx.	~3 million developers worldwide with ~2.2 million apps in the App Store to pull from. Works if you're happy with Apple's finished products.	Overall stable/reliable system. Works well if your target users are familiar with the UX. Can attach brand to Apple's and benefit from a halo effect.
Cons	Minimal design diversity that locks you into the UI, OS, and hardware. Models not available long term so you can't standardize on a particular model at a specific price point, making it hard to build a fleet and maintain compliance and performance.	Prosumer use case focused and not optimized for dedicated devices. Minimal control as you're beholden to Apple's decisions and must work within their constraints. Don't get the granular control you get with Linux or Android.	High price point with minimal range, driving up overall TCO. No alternatives; you're locked in to the prices set by Apple based on their prosumer market goals.	Completely closed ecosystem that Apple defines and controls. Developer pool is focused on what Apple is trying to build with their ecosystem.	Not a way to reach the masses for telehealth or staff use cases as it only has ~30% of smart devices market share worldwide. Performance designed for prosumer use cases, may not stand up to healthcare industry standards. Apple branding will be the focus.

Linux

	Design	Technology	Cost	Resources	End user experience
Pros	Can build any device with any UX you want. Great for leading edge healthcare use cases. Hardware available more long-term because it's IoT focused.	Longstanding OS with a dynamic open source community. Strongest kernel imaginable that's great for high-security verticals like healthcare. Highly flexible infra for app management.	Lots of choice and flexibility. Costs highly dependent on your supplier relationships and the robustness of your design, which you get to control.	Open source, IoT community. Good sized developer pool available to you. If you have an intersection with IoT, this may work for you.	You have complete design control of your UI and branding. Usually see a high level of durability and resilience as this market is IoT driven.
Cons	Few finished devices available here. You'll need to take on full ownership of the design and costs. Can have long lead times so not the best for building off of and going to market quickly.	Focused on IoT world. You won't benefit from the work being done by device or component makers targeting larger consumer and enterprise markets.	Designed for IoT with a thin market for finished devices. Won't know how devices will perform in healthcare use cases as there isn't any data to reference.On your own to navigate the market which doesn't have established prices.	Dependent on being able to find and hire talent as dev pool isn't well documented or followed. You could end up paying developers to build what is readily available to you through other platforms.	Performance is a bit of an unknown for finished devices as data is limited. You are required to take on the burden of designing the UX. Not great for patient or staff facing use cases as few average users know the Linux UI.

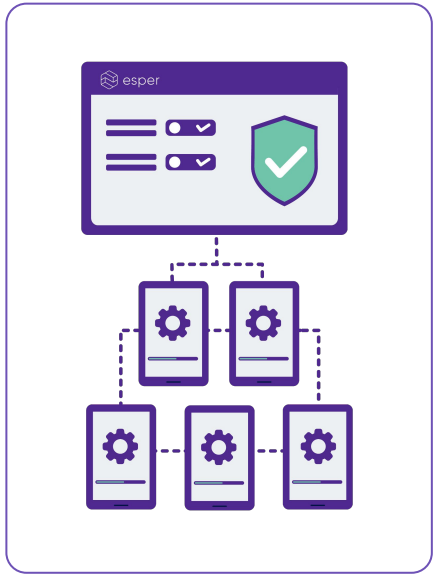
Windows

	Design	Technology	Cost	Resources	End user experience
Pros	PC focused with minimal design experimentation. Some flexibility to design a UX on top of core experience. Some OEMs offer vertical market devices with longer term support.	Long standing OS with good enough security for the enterprise use case. Although not optimized for dedicated devices, it can be used to manage apps on them.	As Windows is a long-standing OS, you're getting known costs. Can find economical option if you're looking at PC as a form factor or compute.	Semi-flexible ecosystem to work with. Tons of Windows apps available, more than iOS and Android combined (although some may be very old).	Older UI can work if you're targeting older users who are the most familiar with it. Can hide some of MSFT's branding from end-users if needed.
Cons	Most designs are for laptops, desktops, and servers, which won't give you a high variety or serve telehealth use cases. Not designed for touch or modern use cases. Committing to MSFT's path and the uncertainty of them investing in smart devices or ARM.	Focused on legacy core IT scenarios such as PCs, not great for healthcare use cases that require kiosks, tablets, handhelds, etc. Tech reaching end of the road based on where the industry is going.	Higher price points with fixed form factors- x86 tends to be more expensive silicon, plus you'll pay licensing fees. Esper has seen many customers move to Android for better TCO.	Locked into stagnant use case as ecosystem is driven by Intel x86 on the Windows OS. Not a huge amount of activity in this pool, especially when compared to the activity in the iOS and Android spaces.	Doesn't stand up well when deployed into the healthcare space. At some point you'll get the Blue Screen of Death. UI wasn't designed for touch and younger users aren't familiar with it.

Android

	Design	Technology	Cost	Resources	End user experience
Pros	Wide range of options for both custom and off the shelf hardware. Can get long term availability or newer models. Lots of tooling to tune and build the type of UX you want.	Flexible and open. Best for executing a diverse set of use cases (i.e. remote monitoring, staff tablets, handheld clinical devices). Can standardize on one OS. Have granular control of apps.	Wide range. Cost mainly driven by hardware choice, which you control. Mid-range options available to find the ideal performance to cost ratio.	Broadest ecosystem. ~6 million developers worldwide with ~3 million apps to pull from for GMS. Can choose dev talent from a highly diverse set of markets.	Most used and known UI worldwide with billions of users. Low barrier to learn UI if patients or staff are unfamiliar or aren't tech savvy.Get full control to tune branding to your device/use case.
Cons	Need to be careful with low end hardware. Do your due diligence and vet the hardware you're selecting. Building custom with AOSP will increase complexity and costs.	Not as customizable as Linux. Will have to navigate wide ecosystem and vet the options available to ensure performance, compliance, and security. Performance dependent on infrastructure and OS version you're getting.	Higher costs for premium hardware competing with Apple. Low cost options might not have the high performance needed for healthcare. More complex decision as you have many options.	Not as wide open as the Linux ecosystem. For AOSP, you typically can't use GMS-based apps. You'll need to work with ISVs to get an AOSP version or build a custom app.	Performance is dependent on OS implementation. Need to do due diligence and test device performance level.

Need more help?



Accelerate your HealthTech innovations with Esper

Esper can help you build the fleet that's right for you, whether you're starting from solution design or transforming an existing fleet.

We believe Android is the best edge device platform for most healthcare organizations. In our experience, it's the ideal choice for fixed-purpose devices because of its flexibility, scalability, security, and ease of use.

Connect with us to see if Android is the right OS for your fleet!

Visit our website or speak with an in-house expert.