Version: 1.0
Updated: 12 15 21

Building a next-generation virtual care platform: A technical guide

twilio

Abstract

The COVID-19 pandemic has accelerated the shift of many companies toward providing care through telehealth and video. The initial telehealth offerings by many providers met the critical need of the moment, but they lacked a clear vision in regard to security, HIPAA-compliance, and user experience. Companies are trying to catch the new wave of what is becoming a booming telehealth industry, and many are considering building their own solution rather than purchasing something off-the-shelf.

In this guide, we'll cover the key considerations and technical requirements IT decision-makers should be aware of in building a modern virtual care platform. This guide will also provide practical steps for technical solution builders to seamlessly create a telehealth platform that fits your organization's strategy. We'll look at the providers' and patients' expectations regarding features. We'll walk through the technology that's required to architect a video chat platform. Most importantly, we'll consider how your organization can use pre-built SDKs and APIs to implement a cost-effective telehealth solution that is highly customizable and delights your users with a memorable video experience.

Contents

Introduction

When the COVID-19 pandemic initially shut down all elective and non-essential, in-person services, health providers jumped to makeshift solutions to facilitate telehealth visits with their patients. The first quarter of 2020 saw a 50% increase in telehealth visits compared with the last quarter of 2019. And we still continue to leverage this technology today.

Yet many organizations have found that their current telehealth platform falls short.

The experience has been less than ideal. General-purpose video conferencing solutions — not built specifically for use in virtual care or integration into healthcare workflows — are insufficient as long-term telehealth solutions. Security and compliance have suffered.

Virtual care is here to stay. As providers shift to thinking about the future for their digital strategy, the conclusion across the board is the same: A solid digital engagement platform is critical to keeping up with patient demand and loyalty.

Why is building your own telehealth platform important?

With the global telemedicine market estimated to reach USD 190 billion by 2025, it's easy to see why many organizations are embracing telehealth solutions. A survey from the COVID-19 Healthcare Coalition shows over 80% of patients reporting satisfaction with the care they've received through telehealth. Given the choice of building their own platform or using an off-the-shelf solution, why are many organizations choosing to build their own?

Organizations want more control over patients' and providers' experiences and seamless integration of their telehealth platform into their existing infrastructure and workflows. A custom-built solution becomes **not just a way to facilitate virtual care, but as a way to differentiate the care they provide**. A state-of-the-art telehealth experience for patients and providers that adapts and scales with your organization makes this possible.

What do you need to know?

In this guide, we'll walk through the key considerations for building a telehealth platform that truly delights your users. We'll look at:

- The core **features of a modern telehealth platform**
- The **technology involved** in building a telehealth video solution
- The key considerations for the **provider and patient experience**
- Practical next steps for **getting started**

Let's start by considering what must go into the modern telehealth platform.

The modern telehealth platform

The core experience of your telehealth platform depends upon if you offer real-time (**synchronous**) interactions, ad-hoc (**asynchronous**) interactions, or both. Let's look at the key differences between these

experiences, some of the platform expectations your users will bring to their telehealth session, and a few advanced offerings that can make your telehealth platform a more "delightful" experience.

Asynchronous sessions versus synchronous sessions

In an asynchronous telehealth experience, communication occurs in an ad-hoc manner — rather than through a live appointment — between the patient and the provider. Patients can use in-app messaging to send text, picture, or video messages to providers when they have new symptoms after a visit. It also provides a solution for non-urgent care.

Synchronous visits, on the other hand, typically occur over video or voice, where the patient and the provider engage in a live discussion. For scheduled telehealth visits, a patient may receive a reminder shortly before the scheduled appointment. The patient may log in to a patient portal, go through onboarding steps, or simply join the video or voice visit at the scheduled time. For on-demand visits, a patient requests to be connected to an available provider. The patient may wait in a virtual waiting room until a provider becomes available.

Minimum requirements for a basic video experience

The pandemic has made video chat a normative experience for the everyday user. Researchers estimate that 81% of Americans have used video calling and conferencing during the pandemic. Contrast this with a survey from Mozilla and Ipsos, in which 38% of respondents had never used a video chat platform before the pandemic.

As it relates to telehealth, this implies that **patients and providers already have expectations for what a basic video experience ought to include**. At a bare minimum, a video chat platform (telehealth or otherwise) must include:

- Device testing prior to joining the video visit, to check the camera and microphone, and to ensure the internet speed will support the video experience
- A "join" button that allows participants to enter the video room
- Basic call control functionality for participants, such as microphone mute/unmute and camera enable/disable
- Screen sharing capability
- Support for text chat alongside the audio/video experience
- A network or bandwidth indicator alerting participants if their internet connection is unstable and if they should turn off their camera
- A speaker indicator that highlights who is currently speaking
- A "leave" button that allows participants to exit the video room
- Option to rejoin video visit through same link if disconnected

Additional features for a delightful video experience

When building a **platform that creates a memorable video experience for your users**, you should consider some of these additional features:

- A waiting room for participants before they are admitted to the call
- The ability to invite a third party to join the video session

- Audio-only sessions in case of limited bandwidth, allowing users to dial in
- Session recording, with secure storage and downloading of recordings
- Remote participant control features, allowing the session host to mute participants or ask them to turn their camera on or off
- A "speaking while muted" indicator
- Custom virtual background options

Advanced features for telehealth-specific solutions

What we've covered so far in the way of features has been related to the general video chat experience, not unique to telehealth platforms alone. When looking at telehealth platforms specifically, additional considerations come to the surface.

Patient waiting room
A virtual waiting room is essential for a telehealth workflow, and it mirrors patients' expectations of the traditional healthcare workflow. However, **a virtual waiting room presents an opportunity to engage with patients** instead of leaving them to stare at a blank screen while they wait. Consider the following ideas:

- **Educational content:** Provide helpful, personalized content — for example, articles or videos related to certain health conditions or healthy lifestyles — which patients can consume while they wait.
- **Estimated waiting time:** If the provider is running late, give patients an estimate for how much longer they might expect to wait.

Invitation for a third party to join
Inviting a family member or caregiver to join a telehealth visit can be valuable to patients. For ad-hoc invitations, SMS is a common channel for inviting a third party with a link to join the session immediately. For third-party invitations to a scheduled appointment, email can be an effective channel for delivering the invitation.

Interpretation service
If a patient and provider speak different languages, or either party relies on sign language interpretation, integrating your telehealth solution with a service that provides language interpretation will be an important differentiator. For example, you could integrate a language interpretation service like VOYCE with a Twilio-based telehealth solution, as described in this instructional article.

EHR/EMR integration
You may consider integrating your telehealth solution with your existing electronic health records (EHR) or electronic medical records (EMR) system. Ideally, you could embed your telehealth app into your EHR, resulting in a seamless integration so that your provider can manage records and conduct video sessions through a single pane of glass. But with EHR integration and resources being limited, you could start with a simple SSO integration for the providers to access a telehealth experience that is technically freestanding and work towards an integration roadmap.

Transfer of health data from Edge devices
With the rise in usage of smart devices to track our activities, **health data such as blood pressure, heart rate, and oxygen levels can be accessed with wearables** and home monitoring devices. That health data can be transmitted from these edge devices to providers during a telehealth session or whenever there may be an anomaly in readings. Taking your telehealth offering to the next level, you might consider designing monitoring devices, making use of Twilio's Programmable Wireless products to deliver IoT connectivity on a global scale.

Customer support
Another feature to complement your full telehealth solution offering is the ability for patients and providers to get technical support when needed (for example, when there's a network connectivity issue or camera malfunction).

A chat widget embedded in your telehealth platform could connect to a support agent directly. If text-based support cannot solve this issue, then you can provide a phone number for your telehealth users to call. There are several ways you can build and architect an interactive voice response (IVR) solution for customer support.

Post-visit patient experience survey and payment self-service
To improve patient experience, consider **sending patients easy follow-up information on how to pay for services and/or feedback survey to improve future experiences**. SMS is a simple and convenient option to send this information. Alternatively, you could build a form that asks for rating and feedback from patients right after they leave the telehealth session.

Now that we've looked at all that could go into a modern telehealth platform, let's focus on the video functionality of the platform, covering the core technology concepts that make it happen.

Technology for telehealth video

When planning to build a telehealth video solution, the first question to ask is this: *Should we build our own solution from scratch, or should we build on top of services, SDKs, and APIs?*

In this section, we'll look at what typically is the technology needed to develop a telehealth solution. Then, see how APIs through a service like Twilio can abstract away those low-level architectural concerns to help make it easier to build your own solution.

WebSocket and signaling

WebSocket provides full-duplex communication channels over a single Transmission Control Protocol (TCP) connection. This enables two-way interactive communication between the web browser for a user (such as a patient or a provider) and a server.

The signaling layer of a video application is important, as it governs how clients interact with one another. This is done through WebSocket. When a user loads the video application, **a WebSocket connection is used to broadcast this user's presence to all other users.** From there, the user can subscribe to the new audio/video stream.

However, WebSocket alone is not typically the protocol used for enterprise-level video streaming between users. For low latency video streaming, a UDP-like protocol is better suited, as there's no overhead to open connections. At the end of the day, slight data packet loss is preferable over a few seconds of delay in a video call.

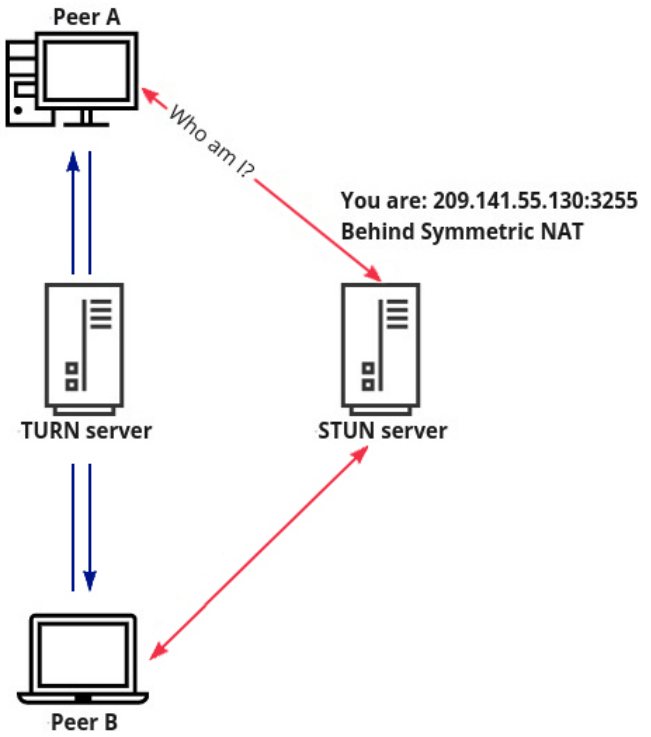
WebRTC

Web Real-Time Communication (WebRTC) allows capturing and streaming of audio, video, and arbitrary data between browsers — without the need to install a plug-in or download any additional software. WebRTC is built around a collection of standards and protocols. Its implementation contains a set of open-source, browser-based APIs supported by most modern browsers. WebRTC has become the de facto standard for building real-time video conferencing applications for use in a web browser, and it boasts several advantages, including low latency, scalability, security, and cross-browser compatibility.

STUN and TURN

Establishing peer-to-peer connectivity for media streaming requires following the Interactive Connectivity Establishment (ICE) standard. ICE includes two main components: Session Traversal Utilities for NAT (STUN) and Traversal Using Relay around NAT (TURN).

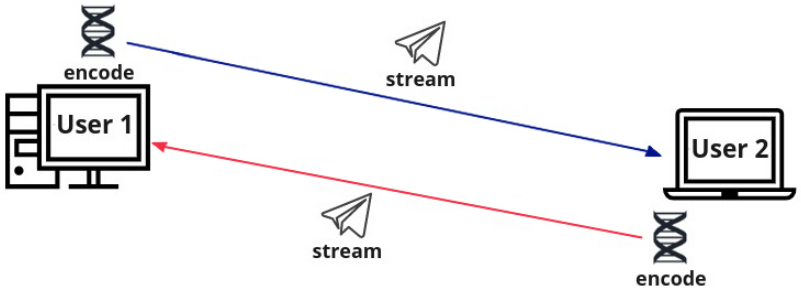
STUN facilitates the identification of a public IP address for clients. STUN servers, which are typically light weight, are deployed to manage this discovery and identification. Once user IP addresses have been identified, signalling can be established between the WebRTC clients. However, there may be occasions when STUN fails to discover the public IP and port of an incoming client request. In these cases, a TURN server is needed to act as a relay to mediate this connection between clients. In short, a TURN server is a STUN server with additional built-in relaying capabilities. As a rule of thumb, around 20% of video sessions can only be established with TURN, whereas around 80% can rely on STUN alone. Companies seeking to build their own video chat platforms would need to manage and deploy their own STUN and TURN servers in order to address these peer-to-peer connection concerns.



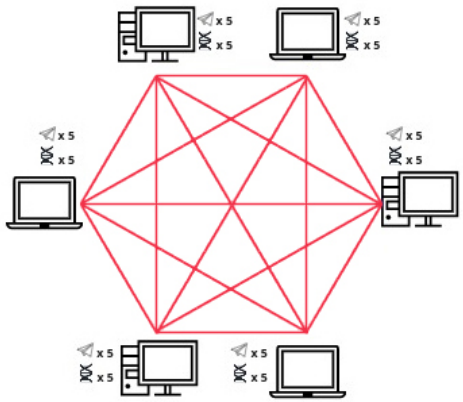
Peer Interactions via STUN and TURN

MCU and SFU

Peer-to-peer communication with WebRTC places an excessive burden on each user's computing and bandwidth resources, especially as the number of participants in a video call increases. For each user added to a call, every other user needs to encode the stream they send to that user and decode the stream received from that user. This approach may be acceptable in a 1:1 video call, but it does not scale well for multi-participant calls.



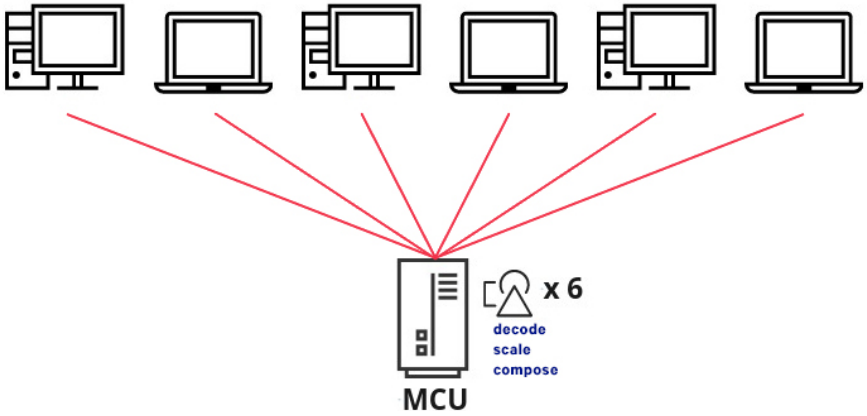
WebRTC 1:1



WebRTC for Six-Person Video Session

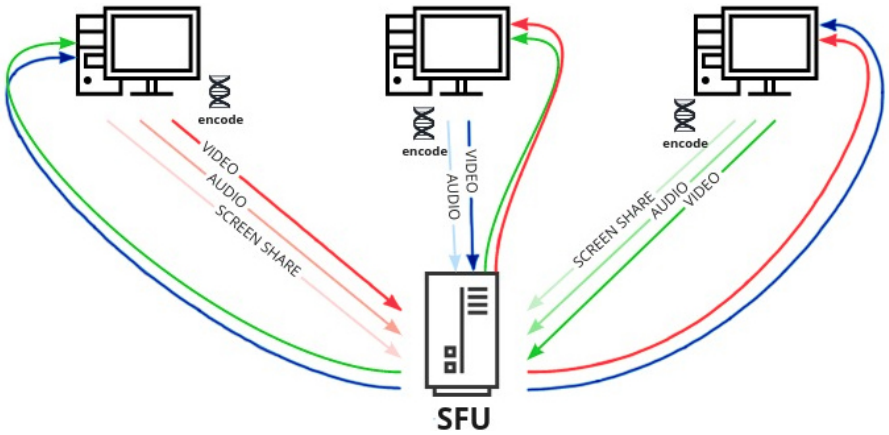
P2P infrastructure

Legacy conferencing systems addressed this issue with Multipoint Control Units (MCU). The **MCU acts as a central point to which each participant sends its media stream.** The MCU handles the decoding and rescaling of each stream. Then, it composes a single new stream to send back to all participants. This way, all participants — regardless of the number of participants in a call — only need to encode and send one stream, and they only need to receive and decode one stream. The main drawback of the MCU is the significant computing power needed to perform its function.



Six-Person Video Session with MCU

More recently, video conferencing platforms have adopted the use of the Selective Forwarding Unit (SFU) over the MCU. In this approach, participants can send multiple streams to the SFU. The SFU chooses which streams to send to participants. The SFU does not handle any decoding or encoding of streams. This reduces its computing needs significantly but subsequently requires more computing power from the end-user, where decoding and encoding will be performed.



Three-Person Video Session with SFU

To build your own platform for telehealth with video, your organization will need to ensure that it has the engineering expertise to implement solutions built on the above technologies as well as the resources to deploy and manage those technologies. This includes:

- Implementation of a WebSocket-based solution for signaling
- Implementation of a WebRTC-based solution for audio and video streaming
- Deployment and management of servers to support WebSocket and WebRTC functionality
- Deployment and management of STUN/TURN servers for peer-to-peer connectivity
- Deployment and management of MCU or SFU servers for multi-party communications

Because of this wide range of considerations — from software development to infrastructure management — typically only medium-to-large enterprises have the human and financial resources for this build-your-own approach. Now let's take a look at how APIs through a service like Twilio can abstract away many of these complexities.

Eliminating architectural concerns with Twilio Programmable Video

Twilio's Programmable Video API lets developers build a video platform solution that **leverages WebRTC while also abstracting away the details of the signaling layer** for simplicity. Developers don't need to worry about these low-level concerns, but rather can focus on customizing their implementation around video session connections and events.

In addition, Twilio's network traversal service provides STUN and TURN capabilities to support WebRTC. Lastly, companies building video platforms with Twilio take advantage of Twilio's Group Room topology, which uses SFU to facilitate low-latency, high-capacity communication between participants in a group video session.

Now that we have covered the main tech stack components for a video platform, let's reflect on some of the key considerations for the user experience of your provider and your patient.

Understanding the provider and the patient experience

As you build out your telehealth solution, you will need to focus on the user experience for the provider and the patient. The typical workflow for the two users may look like this:

1. The **patient** clicks on an appointment link.
2. The **patient** waits in a virtual waiting room until the provider joins the call.
3. The **provider** reviews the patient's chart in the EHR.
4. The **provider** launches the video session (ideally, from directly within the EHR).
5. The **provider** and/or the **patient** can invite a third party to join the visit.
6. The **provider** and the **patient** complete their call and exit the video session.
7. After the visit is over, the **provider** writes session notes in the EHR.

Given this workflow, let's consider both the patient and the provider's experience and environment.

The provider

When developing an application to be used on a computer (rather than a mobile device), you may consider whether you should build a web application or desktop application. Web applications can be accessed anywhere without the need for application or updates. On the other hand, an installed desktop application might not require a constant internet connection for some of its functionality, and it typically optimizes CPU resources with better performance. To help you decide the best approach, consider the following questions.

Will the provider have full access to the internet with a modern browser?
Some hospitals or clinics have more restrictive environments due to security and compliance issues. In this case, a provider may not have access to your web application and may only be allowed to install an approved desktop application.

The desktop application would still need access to the internet with specific IP and port connections available. However, this requirement can be identified and limited to the application by creating specific firewall rules that allow connections to your telehealth desktop application.

How often will a provider use your application — daily, hourly, or continuously?
For providers that are 100% telehealth, the application may need to run continuously through the workday. In this case, a desktop application could be beneficial because it can run all day in the background without accidentally closing a browser window and missing an important notification.

Would it make sense to build both?
Most telehealth applications today are web applications, and building a desktop application alone will likely not be enough to accommodate access for mobile devices like mobile phones and tablets.

One approach is to build a web application first, and then build a desktop application using Electron. Electron provides a wrapper around your web application that allows you to develop a desktop application with technologies familiar to web developers, like HTML, JavaScript, and CSS. Many popular desktop applications today — such as Slack, Visual Studio Code, and Facebook Messenger — are built with Electron.

Building a desktop application with Electron may provide a phased approach to developing your telehealth platform, and it may reduce the level of effort needed for software development.

The patient

When considering the patient experience, the desktop versus web question remains, but with additional nuances. First, most people already have countless mobile applications installed on their phones. Asking patients to download and install yet another application — especially as most don't have telehealth visits that often — will likely not be well-received.

In addition, building and launching a native mobile application can take longer for various reasons, which include:

- Different skills needed between Android and iOS application development
- Mobile operating system requirements to submit, review, and publish mobile applications to the Google Play Store (Android) or the App Store (iOS)
- Variability between capabilities native to each device (i.e. camera, microphone, and geolocation, etc.)

Lastly, for patients who are in areas where high-speed internet is not widely available or prefer to use an audio-only option, you will want to add an optional phone number to dial into the virtual care conference. Supporting this would require using the PSTN (Public Switched Telephone Network) interoperability feature within a WebRTC video room in order to bridge between telecom and the internet, or it would require building a different flow to power a voice-only virtual care option.

For both the provider and the patient experience and regardless of the approach you choose, Twilio Video provides a comprehensive set of tools — including SDKs for JavaScript, Android, and iOS — equipping you to build web and native mobile applications to power your telehealth solution. Twilio also provides a full suite of seamless voice channel capabilities, integrating PSTN voice connectivity directly into video rooms so you do not need to find another service to add a voice-only flow.

While building from scratch is not easy, Twilio Video helps you to stand up scalable video rooms with a single set of tools. These tools not only help you to implement your solution, but also provide what you need to log, analyze, and diagnose issues. Instead of spending time worrying about the infrastructure, your engineering team can focus on what matters to your telehealth solution — features and workflows.

Concrete next steps

Now that we've covered all of the core features, technology, and considerations, it's time to think through concrete next steps. You might be integrating a telehealth solution into your existing workflow or application, or you might be building a brand new, standalone video telehealth solution. With either approach, here are a few good starting points:

Leverage SDKs and APIs from Twilio

The modularity of Twilio's video platform allows you to control every piece of interaction that your providers and patients experience. SDK primitives and REST APIs from Twilio let you integrate seamlessly with your existing application — whether it's white-labeling, UI/UX components, backend services integrations, or database connections. In case you are building a standalone application, consider the open-source React video application as a starting point for development.

Whether you need to build a virtual waiting room — as described in this guide — or build third-party invitation functionality or send appointment reminders, existing SDKs and APIs can smooth out your development path and get you to market faster.

You may also consider a fall-back mechanism for a voice-only call in case video connectivity fails. With this fall-back, your application provides a dial-in conference phone number for both the provider and the patient, or perhaps the application would direct the provider to dial the patient's phone number directly.

Familiarize yourself with best practices for building video

Before development begins and you find yourself increasingly committed to specific approaches or technologies, it would be wise to first become familiar with best practices for building a video chat platform. Developing a high-quality video application requires thoughtful planning. Guides are available, whether you are looking for best practices for building a JavaScript video application for the mobile browser, or whether you are looking for example demonstrations for building on Android or iOS.

Beyond platform-specific information, it's important to consider other important aspects such as:

- network connectivity
- security
- authentication
- room topology
- scaling
- optimization

For more information on these topics, see the "Additional Resources" section at the end of this paper.

Build for HIPAA compliance

When building a telehealth platform, HIPAA compliance is certainly an important concern. Particularly for a video chat platform, HIPAA compliance will have implications for video stream encryption. With Twilio Video, all communication streams traverse encrypted channels and are encrypted at rest. Twilio signs a Business Associate Addendum (BAA) to ensure that the application you build with Twilio SDKs and APIs is configured properly for HIPAA compliance.

Conclusion

Communication through video chat has become the norm, catalyzed by the COVID-19 pandemic. This accelerated the shift in digital communication impacts healthcare profoundly, and telehealth is quickly becoming a normative approach to providing healthcare.

Health organizations are considering how to implement their long-term hybrid care strategy. A scalable and customizable telehealth platform sets up those organizations to face future changes in strategy and infrastructure.

As your organization pivots toward building a video-enabled telehealth platform, understanding the core features and required technology that make up a modern telehealth platform is critical. Once you've considered the provider experience and the patient experience, you can leverage tools like Twilio Video, with its rich set of SDKs and APIs, to build a fully-fledged and memorable video experience for your users.

Learn more at twilio.com/healthcare.

Millions of software developers use Twilio's platform and communication APIs to help businesses build more meaningful relationships with their customers.