The Definitive Guide to

In-App Chat

Table of Contents

Chat has hit the mainstream. In what started as simple 1:1 and group chat, messaging has become the place for real communication. Doctor consultations, classrooms, collaboration... the list goes on. As the innovations flow, realtime online communication embeds itself further in our day-to-day.

Which brings us to this. This eBook will give you a 360-degree view of chat app development, deployment, and scale—and why chat can serve any industry.

The Different Flavors of Chat

Chat isn't just hot air. From telemedicine apps, to customer support, to the realm-wide conversations of online games, there is a vast variety of ways in which we now communicate in realtime online—or, simply, chat. Each of these different ways of using chat has dramatically different features, regulations, and requirements.

Take healthcare. If you're building a healthcare chat app and sending personal health information (PHI), you'd better be ready to build some stringent security safeguards, and complex regulatory compliance measures. At the very least, you'll need to ensure that your app is HIPAA-compliant and that authentication measures are rock solid... all so people can chat.

Or take massively multiplayer game chat, where thousands of users are interacting simultaneously. Scale is everything: there has to be a reliable and steady user experience, complete with individual breakout rooms and channels, all of which run lightning fast and without a break in the action. And with so many users, you'll need machine-learning features like sentiment and language analysis, to analyze and filter out profane messages—and maybe even realtime translation to connect players around the globe.

The latest trend dominating the chat space is chatbots: artificial intelligence-driven systems that can have natural language conversations with humans despite being no more than code—and whose capabilities continue to grow.

From simple, rule-based automated responses, to using deep machine learning and AI, chatbots have appeared in chat apps across the spectrum with the purpose of assisting users in a number of ways. They can search, analyze, filter, augment, and aid based on voice and text inputs, and this functionality is finding its way into applications like customer support, retail, and travel.

With so much variety between apps, it should come as no surprise that there are a number of technologies needed to build the most vital features. Careful vetting should take place to ensure that the solution(s), APIs, and infrastructure you choose to build and power your chat app fulfill your audience's unique requirements, otherwise you're looking at hefty delays in time to market, higher costs, and scalability issues.

The Current Landscape of Chat

When you're building a chat application of any kind— mobile group messaging, multiplayer in-game chat, or customer support and chatbots—choosing the right platforms, frameworks, and protocols can make or break your business.

›››

That's because deciding whether to build or buy a chat app isn't binary. The days of making a decision to do-it-yourself or buy from a vendor are gone.

The question is: how much do I want to spend on building the basics, and how much do I want to spend on building amazing new features by orchestrating existing tools and infrastructure?

Between open-source, IaaS, PaaS, SaaS, SDKs, APIs and microservices, businesses have never had more options for ingredients to build their chat products. And the range of choices is widening at a dizzyingly rapid pace.

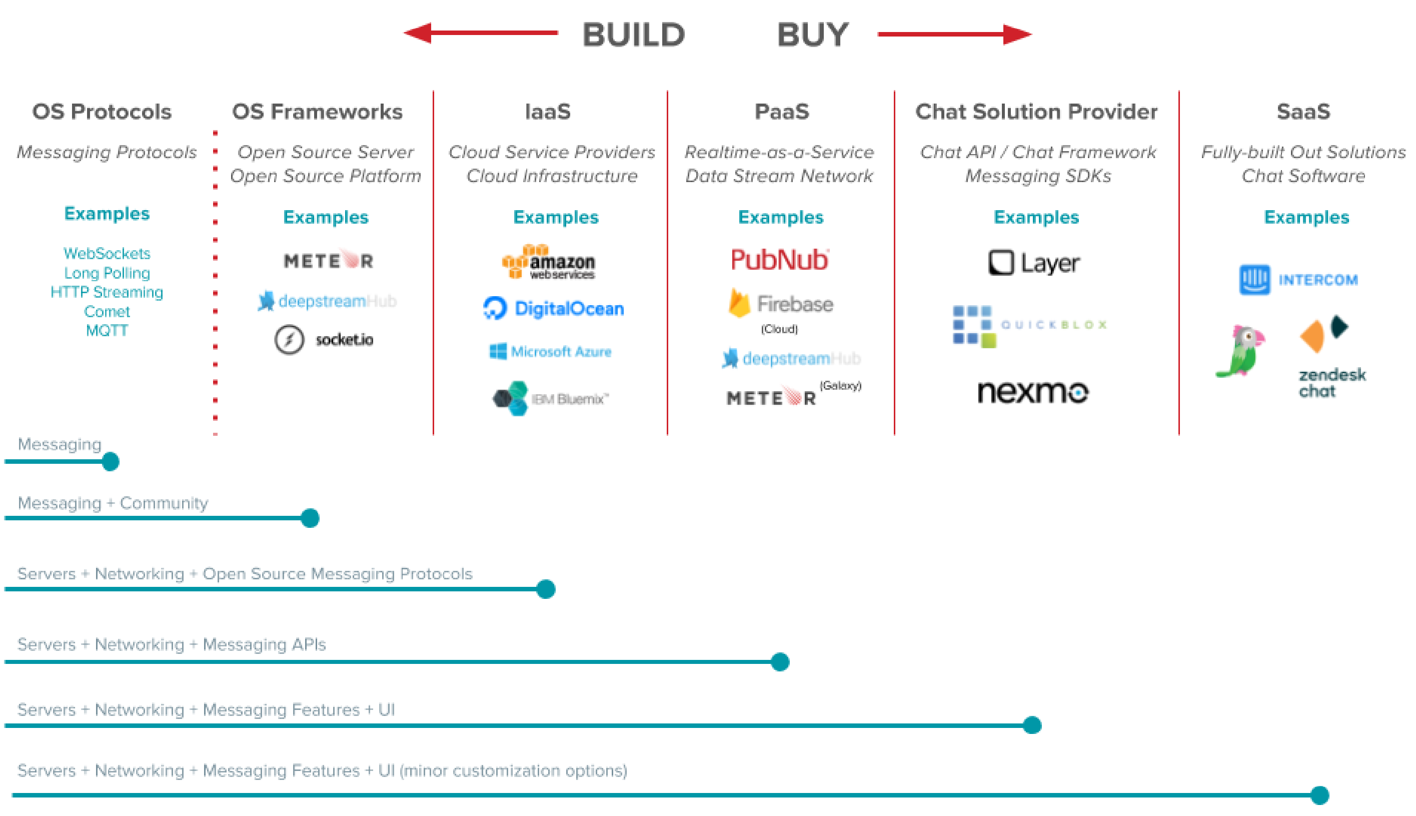
As cloud computing becomes more accessible, predictable, and affordable, innovative companies are working out how to solve specific problems in an entirely off-premise way, driving down costs while improving time-to-market and time-to-innovation. To keep up, development teams are working hard to understand the vendor landscape, along with the benefits and challenges of each option before them.

Developers and organizations can make all kinds of false starts and troublesome early decisions choosing chat or messaging platforms—precisely because the technology is so new, and so broadly defined.

In the next section, we'll discuss a number of different chat application types and look at the different platform options for powering and delivering messaging apps. We'll also discuss challenges that can arise from making certain decisions throughout the development cycle, like scalability, time to market, and time to innovation.

CHOOSING A CHAT SERVICE PROVIDER: CURRENT LANDSCAPE

There is a wide variety of options for chat providers, ranging from the least to the most comprehensive, and the most to the least configurable. You have open-source on one end and fully built out (SaaS) solutions on the other. With hundreds of options in between, all with different pros and cons, the following chart provides a simple way to categorize the marketplace:



OPEN-SOURCE PROTOCOLS

The furthest on the build side are open-source protocols like WebSockets and HTTP Long Polling. These are simply protocols, which means that you manage everything to make them work. That includes spinning up your back-end infrastructure, maintaining it, building new SDKs to support new devices and languages, and everything else.

These are great for prototyping, building small applications, or getting your hands dirty with the full stack. The challenge will come when you move to Production: unexpectedly large spikes in traffic can overwhelm the connection maximums for your server, unexpected security holes can put user information at risk, and you'll need to keep users connected when they jump from network to network—the unknowns are many and deep when going it fully alone.

OPEN-SOURCE FRAMEWORKS

Open-source frameworks are a step past pure build, and of course allow you to build and maintain the infrastructure on your own. Open-source frameworks tend to rely on a community of developers to update the framework and maintain the client SDKs, so what you see when you start is what you can rely on—think of this as an environment with a 'distributed roadmap'.

The flexibility and configurability are still high, as is the list of unknown unknowns that lurk in the future of the application.

INFRASTRUCTURE-AS-A-SERVICE (IAAS)

These solutions are offered by the big dogs—cloud infrastructure service providers like AWS, Digital Ocean, Azure, Bluemix, and Google Cloud. They actually end up powering a lot of the PaaS messaging solution providers, as well as the SaaS products that we'll talk about next.

In a nutshell, you can use open-source protocols with an IaaS to launch your app. The infrastructure is taken care of, from a pure 'the-network-is-up' perspective, but you'll be responsible for the realtime-specific elements that distinguish chat: persistent device connections, security, distributed database replication, global server load balancing (between cloud regions) and so forth.

PLATFORM-AS-A-SERVICE (PAAS)

PaaS providers like PubNub and Firebase offer hosted solutions for building chat applications. They include not only the infrastructure, but also the SDKs, APIs, and integrations for building chat features. Building and customizing the application requires engineering resources, as flexibility is still high—and arguably higher, as pre-built integrations to services like language translation, mapping, and alert gateways accelerate the process of releasing a unique bundle of features.

The greatest benefit, however, is in the chat-ready infrastructure: developers can offload the security requirements and maintenance of the service (the back-end and the client SDKs), as these are handled by the PaaS. Meanwhile, they still enjoy the uptime and support benefits of IAAS.

CHAT FRAMEWORKS

Framework providers are as close to buying a pre-built solution as you can get, while still offering the opportunity to distinguish your offering through unique features that do, in fairness, require a fair amount of engineering. The big difference between these providers and PaaS is that they provide more of a black box approach—you have less flexibility to customize the APIs, the infrastructure, and, to varying degrees, the UI.

Infrastructure will for sure be a black box—so no headaches scaling or maintaining security, in principle. It is important, however, to understand the underlying technologies and network topology to ensure that the proffered infrastructure is built appropriately for your specific use-case.

SAAS

Lastly, furthest over on the buy side of the spectrum we find SaaS companies, who provide a fully-built out solution that requires only a very small amount of engineering. UI, integrations, and infrastructure are all handled by the SaaS provider. Leaders in the space include Intercom, Olark, and Zendesk Chat.

These solutions are ideal when you have a discrete problem to solve, in which the uniqueness of your solution (beyond the cosmetic) is relatively unimportant. Adding a salesperson chat widget to your website, for instance, yields the best results when all the expected features (and no more) are in place, and extra engineering is likely wasted effort in most cases.

QUESTIONS TO ASK YOURSELF WHEN CHOOSING YOUR CHAT SERVICE PROVIDER

As with all the other parts of critical infrastructure, the key questions are going to be:

1. Should you run your own infrastructure, or utilize a hosted service?
2. How much does your chosen approach cost upfront? How much will it cost at scale?
3. Is any hosted service you are considering reliable, secure, and scalable? What guarantees are they willing to provide?
4. How mission-critical is chat to your application?
5. Who on your team will maintain the system? Do you have the skills in-house already to make the system scalable and secure?
6. Where does the service store data, and who has access to it?

CHOOSING YOUR CHAT SERVICE PROVIDER: OPEN-SOURCE VS. HOSTED

When it comes to software development, everyone knows that what works in the lab is not guaranteed to work in the wild. That's because the wild presents unpredictable challenges that no sane developer can dream up before their product gets into the hands of unconstrained consumers.

When it comes to choosing the right technology to power your chat, there are a number of build and buy considerations to consider. For the sake of brevity, in this section we'll just look at the security, scalability, and reliability of licensed infrastructure.

INFRASTRUCTURE

If you're going down the open-source route, you'll choose your tool, install it, and orchestrate the operation of that tool.

From there, you'll start thinking about the infrastructure side of things, like load-balancing and redundant nodes. These are requirements for launching an app at scale. This is when you may tap an IaaS provider to handle the back-end. Even so, it will still require heavy engineering, including:

- Spinning up multiple testing, staging, and production environments
- Coordinating provisioning for those multiple environments (from straight-up rack-and-stack in a data center to Kubernetes containers)
- Deploying your application code to the environments
- Setting up service management, system monitoring, and Ops alerting
- Creating a load balancing scheme (like Nginx or HAProxy)

- Implementing a scheme to segment data by channels or topics (like Redis pub/sub with Socket.io)
- Finding a store and forward solution for signal recovery, like in-memory caching
- Implementing a method to detect connect individual clients to the ideal data center and port (broadly speaking, global server load balancing)
- Computing which channels/topics to send/receive for a given client
- Building orchestration between data centers/cloud regions to ensure data reliability between endpoints
- Deciding which platforms and languages to support
- Creating universal data serialization
- Customizing code to detect data uplink that works across device types
- Determining Quality of Service and level of loss boundaries, and developing a data recovery scheme

That's a laundry list of considerations, of course, and it is not even truly comprehensive! When choosing the open-source route, however, these are some of the upgrades that have to be built in order to transition from lab to production.

SECURITY

For chat, security is critical (nobody wants a busybody looking over their digital shoulder) and tricky. Users are increasingly sending more confidential and mission-critical information via chat applications, from financial details to chatbot commands. Ensuring that you have full control over access and encryption has become table-stakes.

Every successful chat service provider offers different levels of security.

Here are the most important features that must be included in any hosted-service provider:

- End-to-end encryption with TLS for in/outbound packets and AES for packets
- Fine-grained, token-based access control. Token-based access control allows you to grant and revoke access to any messaging channel.
- Regulatory compliance support for any app that exists in a regulated environment. The hosted-service provider should be certified for HIPAA (healthcare), SOC 2, GDPR (EU), Data Shield, and SafeHarbor (EU/US).

For those who choose not to utilize a hosted-service provider, the following are additional security considerations that you'll have to handle on your own:

- Managing TLS, from certificate purchase and renewal, to implementing methods to effectively utilize it in chat interaction
- Figuring out how to protect channels and topics (not covered by TLS)
- Building an authorization system for users
- Considering AES and/or RSA encryption for payloads (not covered by TLS)
- Complying with legislative security policies (like SafeHarbor or HIPAA)

SCALABILITY

For chat apps with thousands of active users chatting simultaneously, and ones that continue to grow, building infrastructure for scale can be a major challenge, not least because the nature of the application is to experience uneven traffic patterns. Both open-source and some hosted-service providers deal with scalability to some degree, but overall, hosted solutions will more completely mitigate the risk of app-breaking scalability issues than than open-source options.

For hosted solutions, there are a couple indicators that your service of choice will scale with your app growth:

- **Multiple global points of presence:** Chat messages should be globally replicated, so that if messages are dropped, a backup message can be delivered. This also increases the performance of your application, as every chat user doesn't have to connect to the same data center (especially those halfway across the earth).
- **Uptime SLAs:** Uptime SLAs hold hosted-service providers accountable, and you should receive credits when SLAs are not met.

For the do-it-yourselfers, you need to consider:

- A custom-built load testing service that can simulate a realistic audience
- Creating update protocol & continuously modifying your network to support new products/services
- Paying for Socket server costs, QA systems, and hot failovers
- Ongoing Ops monitoring and additional headcount required

RELIABILITY

There is a heated competition for messaging applications: with the app store a click away, any issue a user encounters can lead them to an alternative. Reliability is a key factor in making your app sticky. When vetting hosted service providers, here are a couple key indicators of reliability:

- Data replication for multiple points of presence and automatic failover to ensure that messages are delivered 100% of the time (and actually in realtime)
- Message "catch-up" in case of connection dropout (if a user is in a tunnel, for example, they'll receive the message when they come out the other side)

If using an open-source solution, you'll have to also handle:

- Building a load distribution system
- Identifying error messages
- Building a log system
- Knowing when faults occur and developing a playbook of responses
- Building service management (like PagerDuty)
- Developing multi-datacenter deployment

OPEN-SOURCE VS. HOSTED

Building out a realtime messaging system on your own poses a lot of risks, particularly as the use case becomes more unique, and the audience larger. Going it alone can be a great option for smaller chat applications, but as the audience starts to grow, security, reliability, and scalability challenges can add up.

Most hosted-solution providers also allow a free-forever sandbox pricing tier. This allows you to develop your app without paying, only breaking out the checkbook once you reach critical mass. This gets your product to market, and in front of consumers, quickly and at very low risk. For those companies looking to move fast and not wanting to worry about all the intricacies of networking and infrastructure, hosted-solutions are the way to go.

Chat of the Future + What Users Crave

Messaging apps are growing more innovative, more powerful, and have become a linchpin for customers, businesses, and teams across every industry and vertical.

›››

We've seen a flood of new messaging applications of all shapes and sizes entering the market. In fact, messaging apps have surpassed social networks in monthly users, and the gap continues to grow. More and more functionality is taking place directly in the chat interface outside of sent and received messages.

So how do you set yours apart from the rest?

Obviously, you'll need a certain set of core, established features that users have come to expect: presence detection, typing indicators, reactions, that sort of thing. These are table-stakes features.

Introducing a new app, though, is normally about taking something to the next level, differentiated from the thousands of chat apps already out there. Or, it's about building messaging into an existing application to deliver a better user experience and drive engagement. By combining the vast number of incredibly accessible, affordable, and easily-available APIs, SDKs, integrations, and infrastructures-as-a-service, you can do just that.

In this section, we'll start with features every user expects, then move into innovative ideas for building the chat of the future.

EXPECTED FEATURES OF ANY CHAT APP

These are the fundamental features that every chat has (or should have) today:

- Send and receive messages in realtime, either one-to-one, or one-to-many.
- Join and leave a chatroom.
- Messages should be stored and easily retrieved, either through a Load more messages option or infinite scroll.

- Messages should be time stamped.
- A user list, both public and private, that updates in realtime as users go online/away/offline.
- Typing indicators to show when someone is actively typing.
- Alerts for offline users via push notification when they receive a message.
- Only authorized users to have access to channels to keep the conversations private when necessary.
- Ability to invite other users to a group or remove/block them for inappropriate actions.
- .gifs. So many .gifs.
- Emoji and reactions 😬

CHAT FEATURES TO SET YOURSELF APART

Obviously, we're all very familiar with the features listed above. Give or take, they're at the heart of basically every messaging application out there today.

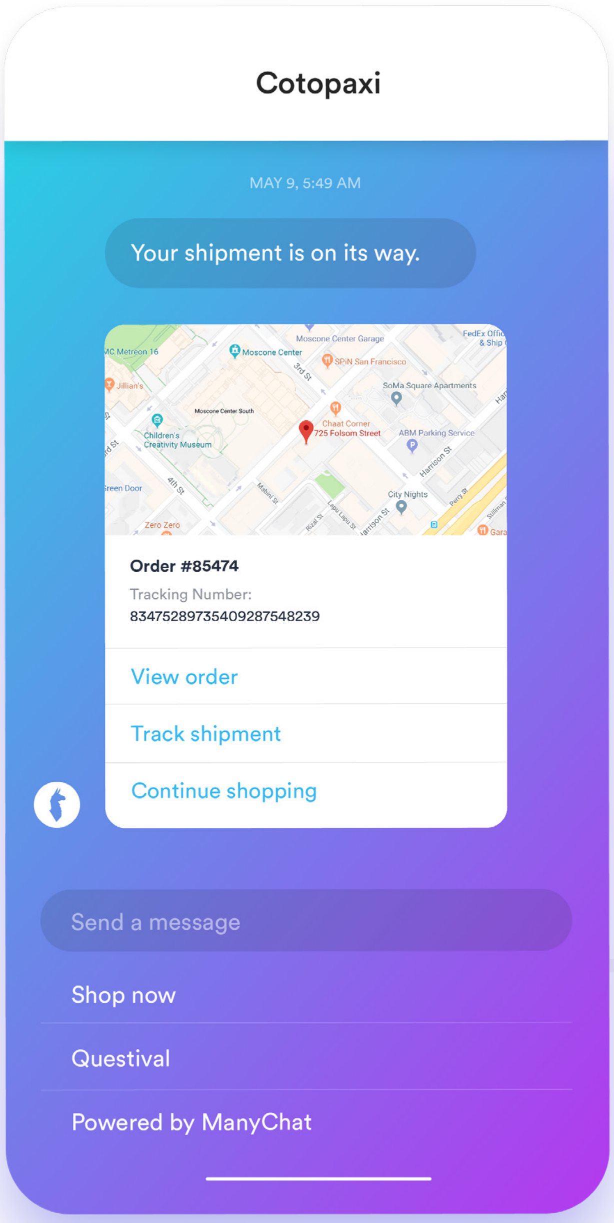
More interesting are the emerging technologies allowing you to build the features of the future, which will set your messaging app apart. These kinds of features are redefining the way communication takes place, online, offline, and everywhere in between.

PROGRAMMABILITY

Before getting into specific differentiators, it's important to first discuss how important it is to build your chat application with programmability in mind. Programmability means your chat app is "event-driven", giving you complete control over everything that happens with a message, from the moment it is conceived to the moment it is received. This could mean adding realtime translation, alerts, or triggering 3rd party APIs. This design pattern drives the sorts of innovative features listed below.

GEOLOCATION

Instead of forcing users to switch between messaging and mapping apps, deliver geolocation directly to chat users.



MAY 9, 5:49 AM

Your shipment is on its way.

Order #85474
Tracking Number:
834752897354092875482 39

- Embed live maps with realtime geolocation tracking directly into a message stream. eCommerce and on-demand companies can utilize this to allow customers to monitor their delivery or service in realtime.

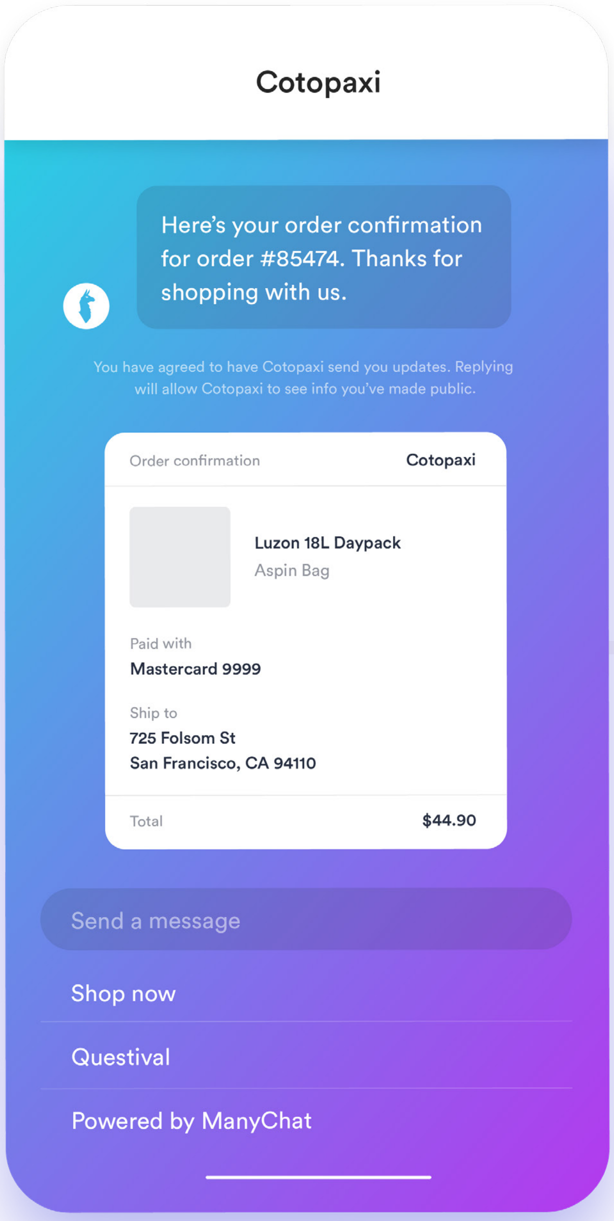
  Integration examples: Mapbox, Esri, Google Maps

- Initiate directions and ETAs and publish them directly to the message stream. Combine with text-to-speech to get audio directions, and provide the option to publish progress as users travel—it's the best way to know how far away the birthday cake, surprise party honoree, or long-lost cousins really are.

  Integration examples: Mapbox, Esri

ECOMMERCE AND ON-DEMAND

Run an entire shopping experience, from browsing to fulfilling orders with chat.


Cotopaxi

Here's your order confirmation for order #85474. Thanks for shopping with us.

You have agreed to have Cotopaxi send you updates. Replying will allow Cotopaxi to see info you've made public.

Order confirmation Cotopaxi

Luzon 18L Daypack
Aspin Bag

Paid with
Mastercard 9999

Ship to
725 Folsom St
San Francisco, CA 94110

Total $44.90

- Utilize chatbots with automated and AI-powered cognitive capabilities to assist shoppers with browsing, ordering, and troubleshooting orders from a messaging interface.
- Stream order updates and alerts, like push notifications and SMS, as the order moves through the fulfillment lifecycle.

  Integration examples: Clicksend (SMS), Infobip (SMS), RingCentral (SMS), SendGrid (Transactional Email)

MODERATION

There are a lot of trolls out there. Chat apps today need moderation tools in them to filter messages on the fly, not only for text, but also for images.

- Moderate content flowing through a chat as the messages are published on the fly. This could filter out inappropriate messages, block trolls, or even remove competition mentions (for live events and launches).

  Integration examples: SiftNinja, Neutrino

- Moderate images on the fly using image recognition to filter out provocative images.

  Integration examples: Clarifai, Sightengine, AICeption

LIVE EVENTS

Chat can complement a video or live event, and allow users to interact with one another.

- Combine a chat application with a live streaming video, and show how many users are currently viewing the live event.
- Collect and aggregate geolocation data of users to gain insight on how many people around the world are watching the event.
- Allow users to vote on custom polls to increase user engagement.
- Add interactive features like allowing users to raise their hands or upvote other messages.

USER EXPERIENCE

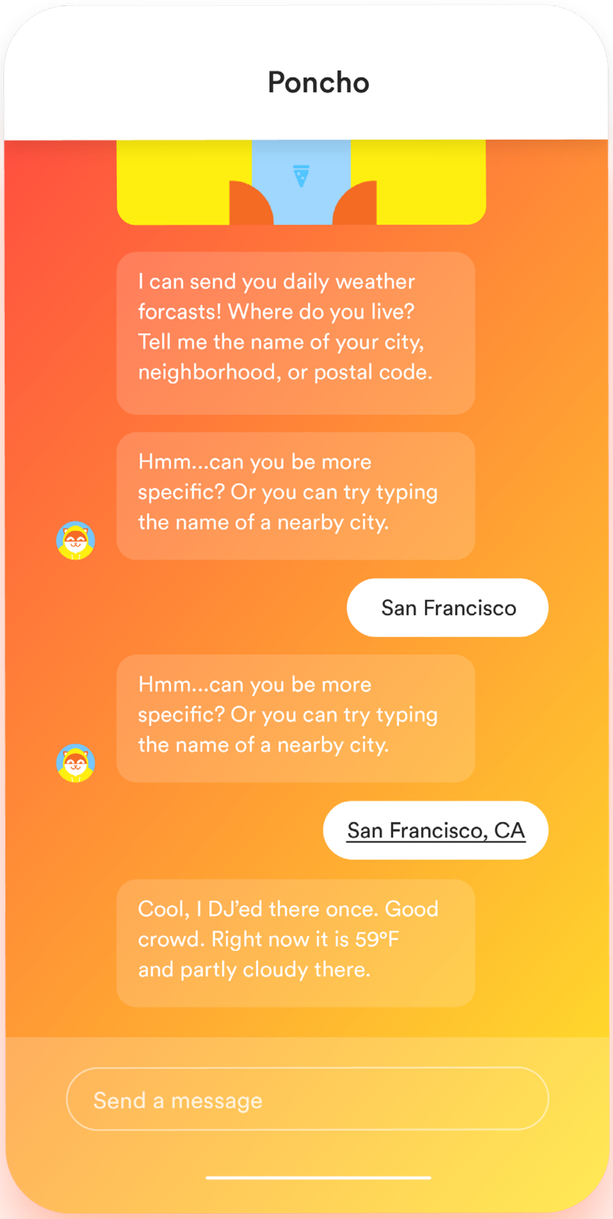
Build delightful user experiences for users.

- Send larger files of any format in realtime. Rather than sending the entire file, efficient chat applications send a reference to where the file is hosted for the end user to download.
- Users should be able to bookmark or star important messages and easily retrieve them later.
- Message search and message tagging/pinning.

AUTOMATED AND AI-POWERED CHATBOTS

Chatbots are difficult, but, luckily for you there are a ton of powerful chatbot services on the market today. You should be able to integrate all sorts of chat microservices to power your chatbots directly into your application.

- Use automation to send responses when a situation is predictable. For example, a user starts chatting with a support agent on a website. While an agent may not be available immediately, you can


Poncho

I can send you daily weather forcasts! Where do you live? Tell me the name of your city, neighborhood, or postal code.

Hmm...can you be more specific? Or you can try typing the name of a nearby city.

San Francisco

Hmm...can you be more specific? Or you can try typing the name of a nearby city.

San Francisco, CA

Cool, I DJ'ed there once. Good crowd. Right now it is 59°F and partly cloudy there.

collect the user information, like email, through automated responses. This benefits the user as they won't have to wait for long to hear back from an agent and frees up time
for agents.

- Use automation to send responses when the data is available and accessible. In the example of an eCommerce website, the user can type in a chatbox, "I would like to know the status of the refund for my order." The intent of this sentence is *"refund status."* An automated response could be sent to the customer with a mini form asking them to provide their order ID and email. This eases the user data collection experience.
- We can make users' experiences better by asking questions and giving automatic responses contextually depending on where the user is on the site. While they are on the payment page, for example, you can show them a personalized coupon.
- Combine activity history with customer inputs to make recommendations and answer questions in realtime.

THE SKY'S THE LIMIT WITH INTEGRATIONS AND APIS

- The number and range of 3rd party services now available to pull into your application are truly staggering. Manipulating a combination of these services is the surest, safest, and quickest path to creating something truly unique, that solves a particular user challenge completely. The more unique offerings will combine these technologies to take the market by storm.
- Interact with 3rd party apps and APIs using commands (like Slack apps) to make your tasks more efficient.
- Utilize 3rd party alerts and notifications services to trigger SMS, mobile push notifications, and browser notifications.
- Initiate a video call or interactive collaborative environment with services like Cisco Spark.
- Create your own API which can be called from chat using a command that fulfills your needs only.

ANALYTICS

You should give your business users full transparency into the usage of their application.

- For customer support types of messaging apps, allow business users to see average chat time, connected end users, open and closed chats, or even sentiment ratings of conversations (using APIs like Watson Natural Language Processing).
- Monitor geolocation of connected users to gain a better understanding of where your users are located.

Conclusion

The world of chat is big, and it gets bigger and more complex every day. As your organization seeks to dip its toe into the waters—or to expand your influence and dominate—you'll need to make key decisions on development environment, delivery and scalability options, and the customer-facing features that will set your product apart. You now know how to balance the pros and cons, and are doubtless already plotting your next great product.

Make no mistake: in-app chat is the future of communication.

PubNub

www.pubnub.com

(415) 223-7552

@pubnub

Contact Us