ROOKOUT

# Engineering Leaders' Survival Guide to Solving Customer Issues

As an engineering leader and co-founder at Rookout, nothing's more frustrating than a customer issue escalated to engineering. Whether it's a new or existing customer, the business has already suffered a setback, making their job harder. And now, engineers will squander precious working hours chasing after the issue. If everything goes well, we will hopefully be back on schedule, but you never know what you will find as you dive into the rabbit hole of customer issues.

At Rookout, resolving those escalated issues as fast as possible while delivering our customers a delightful experience is a top priority. I want to take this opportunity to share with you some of what we learned on how you can deal with those issues fast and deliver your customers a superior experience.

> ***Nothing's more frustrating than a customer issue escalated to engineering.***

**KEY TAKEAWAYS:**

Resolution of customer issues is the most dominant form of unplanned work.

Take it a step further, and escalated customer issues can be a huge drain on R&D resources, negatively impacting velocity, technical debt, and employee motivation.

When looking at it from the business side, this is usually the most important and urgent task.

There are five key steps to managing customer issues and ensuring that your company isn't negatively impacted. They are: preparation, prioritization, improving the product, empowering the business, and empowering the engineers.

## Prepare Your Team

First and foremost, you shouldn't be surprised when those customer issues come calling. After all, the only way to avoid having customer issues is to have no customers. Things are different when dealing with an existing application versus one that's brand new.

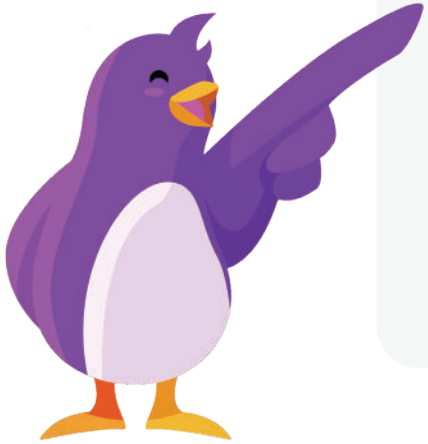
When managing an existing application, you have a history of escalated support issues. So go ahead and be data-driven. Here are some of the questions you should be asking yourself:

1. How many escalated issues should you expect every month?
2. What kind of escalated issues should you expect?
3. How long will each escalated issue take to resolve?
4. What was the business impact of past issues?

Based on those answers, you need to allocate a portion of your team's time to handle those escalations. Keep in mind that you will want a mix of different engineers - frontend and backend, junior and senior, based on the types of problems you have seen. You need to align resource allocation with the business side of things, whether sales, customer success or support. Together, you have to decide how many resources should be diverted from future development to support existing customers.

When dealing with new software, especially one that is just launching, things are less predictable. You may not know when the issues will start coming in, how many there will be, or how to handle them. Depending on how you will launch the software, it may come as a trickle or as a flood.

In those cases, the most important thing is to set expectations. Your team has been busy developing new features, but once customers start coming in, they will have to double down on triaging issues and fixing bugs. You may not know what kind of problems to expect, how to handle them, and how to communicate them to the business team. Make sure to assure them it's a priority for you.



> ***Whether you have to decide which cases to fix and which not, or if you just have to decide which tickets to handle first, effective prioritization is crucial.***

## Prioritize

Rarely will you find yourself facing a single customer issue. More often than not, you are going to have a long list of tickets in front of you and quite a few stakeholders needing them resolved as soon as possible. Whether you have to decide which cases to fix and which not, or if you just have to decide which tickets to handle first, effective prioritization is crucial. Whenever prioritization comes into play, you must manage two key aspects: the criteria and the process.

When I find myself sorting through customer support issues, there are four main perspectives to evaluate. For the most part, prioritize **high business impact** and **mature** issues that align well with the **product and tech roadmap** and are **affordable** to resolve. Of course, nothing is ever easy, and many cases will score high in some parts and low in others.

Here are some examples of the questions you should be asking yourself:

1. **Business Impact** - How important is the customer to the company? Are there significant commercial agreements in the process? How much of an impact does this issue have on the customer and the relationship? Except for this issue, how well is the customer satisfied with our support?
2. **Issue Maturity** - How well is the issue defined? Did the support, success, professional services, and solution engineering teams exhaust their options? Do we have all the information needed for engineering to work to resolve the issue effectively?
3. **Product and Tech Roadmap** - Is the issue a bug report or just a feature request? Does the issue affect a high-priority part of the product? Regardless of this specific customer issue, do you want to invest resources in that component?
4. **Resource Cost** - Will this issue be fast to fix? Do you have a precise estimation of what is required? Do you have the relevant personnel available to work on it?

When it comes to priorities, well-defined criteria are the easy part. When revenue (and even individual employees' compensation) is at risk, tempers may run hot. More than anything else, you need to have a well thought out and agreed upon process to keep all the stakeholders aligned and committed to the decisions. The process should start with assigning someone to be responsible for keeping track of all the customer issues in the relevant (part of) the product. This person should probably be a product manager or a tech lead. They should be the point of contact to which all customer requests and help desk tickets requiring engineering involvement flow.

Once you have an apparent backlog of customer issues, you need to have a forum to discuss and set priorities. The group should include engineering and product as well as business stakeholders. The appropriate business stakeholders will change from organization to organization but may consist of customer success, sales, sales engineering, professional services, support, field product, etc. I highly recommend having that forum meet face-to-face (these days, this mostly means video conference) at regular intervals.

Last but not least, the process needs to include relevant exceptions. Who has the final vote on prioritization? How do you prioritize an urgent issue before the next meeting? How do you deprioritize a case if it's taking too long or if circumstances change?


## Improve the Product

As you are going through the tedious process of resolving customer issues one by one, you must not lose sight of the big picture. You must always think of support as both an opportunity and a threat.

Support is a threat because your customers are there to use and benefit from your product. For the most part, as soon as they have had to ask for support, they are not satisfied. Even worse, you'll find many customers who don't ask for help in the first place. Simultaneously, as you keep rolling out new features and products, expect every one of them to generate further support requests. If you are not careful, support efforts may spiral out of control, drowning in new feature development.

However, support is also an opportunity because it provides powerful knowledge to improve the product. By continuously improving the product, you will not only achieve higher customer satisfaction, but you will also be able to control and minimize unplanned work due to escalated customer requests.

Keep in mind that learning from customer support issues usually falls in the product management domain. There are, however, certain classes of support tickets that are of particular importance to engineering managers:

1. **Bug reports** - you may find specific components receive a substantial amount of bug reports. If that's the case, you have a quality problem on your hand. Improve quality through more rigorous testing (including user testing!), higher code quality, and re-writing parts of the code in rare circumstances.
2. **Manual operations** - you will often have your engineering team tasked with carrying certain manual operations. For example, setting various account configurations or running some database operations. If those are becoming an issue, consider exposing certain functions to the in-house support or the customer as part of the product roadmap.
3. **Similar feature requests** - occasionally, your team will develop related features on a case-by-case basis for individual customers. If that happens to be the case, you will benefit from developing the capability in a generic, widely-applicable way. Not only will it provide additional value to your customers, but it will also save your team the necessity of ad-hoc implementations time after time.

> ***Techniques such as non-breaking breakpoints empower your engineers to operate in customer-facing environments safely.***

## Empower the Business

Personally, when dealing with escalated customer issues, nothing is more frustrating than that feeling that the customer facing team should have fixed it themselves. The feeling that by bringing the problems to the teams' attention, they have somehow failed at their job. Unfortunately, I usually find I have no one to blame but myself. It falls upon engineering and product leadership to empower the business team to handle customer issues in the best way possible.

The first thing they should do is to **triage** the issue. Questions such as what's the customer trying to accomplish, why is he doing that, and what's wrong. They should correlate those answers with a **body of knowledge** to better understand the problem and which part of the application is relevant.

Based on that initial analysis, they need to **collect** the required **information**. Such information will vary depending on your product's nature but may include anything from the operating system and browser configurations to cloud providers and third-party vendors. It's your responsibility to define and communicate the information that is needed. If possible, ensure that information is collected and stored using the business IT systems such as CRM.

Most importantly, you want the business teams to **resolve** as many issues as possible on their own. First, they need to have as many scenarios and solutions as possible within that **body of knowledge**. Second, they need various back-office **tools** to carry out configuration changes by themselves. Keep in mind that the more you can automate this process, the more likely it will resolve it without escalating to engineering.

## Empower your Engineers

No matter how good you become at collaborating with your business partners, you will end up assigning escalated issues to your software engineers. While they will (hopefully) get some initial context about the nature of the problem, they will probably have to get their hands dirty.

Software engineers are the masters of your product's source code, and that's where they will dive in to find the solution. Unfortunately, software engineers have traditionally been shunned from customer-facing environments such as production. Most companies keep those environments tightly locked down to control operations risks such as security, availability, and performance.

Few things are more demotivating than hunting for a bug you can't even see. Assuming engineers have the access they deserve to logs and APM tools, those provide minimal visibility into the business logic. Reproducing bugs in a lower environment tends to be a very time-consuming process, especially when flows are highly-customizable or event-driven. Most of our customers report they used to spend much more time getting the issues to reproduce in an environment where they can operate than they do in actual debugging.

Your engineers deserve better tooling to handle those escalations. Techniques such as non-breaking breakpoints **empower** your engineers to operate in customer-facing environments safely. They can see the application code as it executes in the relevant users' context, observing specific variables, and stack-traces as needed. Rookout manages all operational risks such as security, availability, and performance using a policy-based approach.

## Summary

As engineering leaders, few parts of our roles create as much friction as escalated support issues. While it rarely feels like we'll ever reach the top of the summit, it's a very long fall to the bottom. We must make sure to partner with product management and business leadership and manage it appropriately.

By **preparing our team**, creating clear **prioritization** criteria and process, continuously **improving the product**, working to **empower the business**, and most importantly **empower our engineers**, we can support our **growth** while maintaining our **feature velocity**.