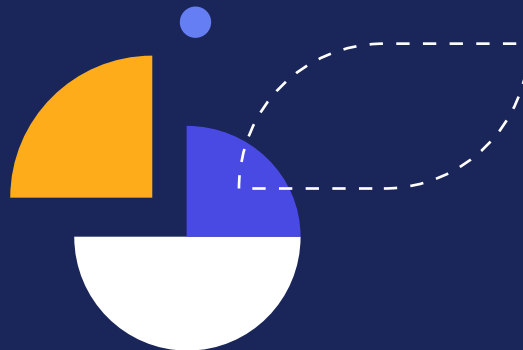




BENCHMARKS

Application Stability Index: Are your apps healthy?



Stability is the measurement of application health and user experience. It is usually calculated in two ways to provide the complete picture — as a percentage of application sessions that are crash-free or did not result in an unhandled exception and as a percentage of daily active users who do not experience an error.

$$\text{Stability Score} = \left(1 - \frac{\text{crashed user sessions}}{\text{total user sessions}} \right) \times 100$$

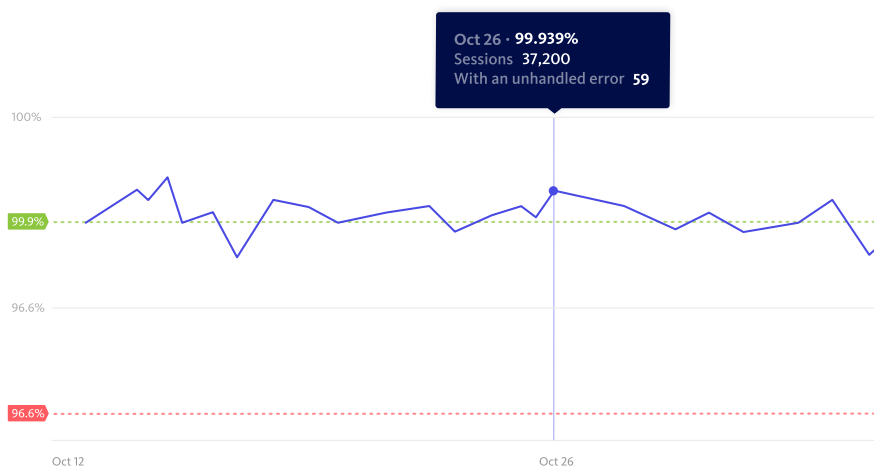
How many nines of stability do you have?

Providing an error-free experience to your customers is of utmost importance to drive conversion, engagement, and retention. Although stability is commonly a KPI owned by engineering organizations, it has a significant impact on overall business performance and growth.

To help you understand how your application compares, Bugsnap has compiled data on the session-based stability of leading mobile and web applications from various market segments, including eCommerce, media and entertainment, financial services, logistics, gaming, and more.

This data can also help you determine application stability SLAs and SLOs, which are similar to the “five nines” that infrastructure and operational teams use to measure uptime and availability. This metrics-driven approach can help development teams make decisions about when to build features vs. fix bugs based on the application’s current stability.

84% of users
abandon an
application after
seeing two crashes¹



SLO → TARGET STABILITY

- Aspirational goal
- Balance fixing bugs with building new features

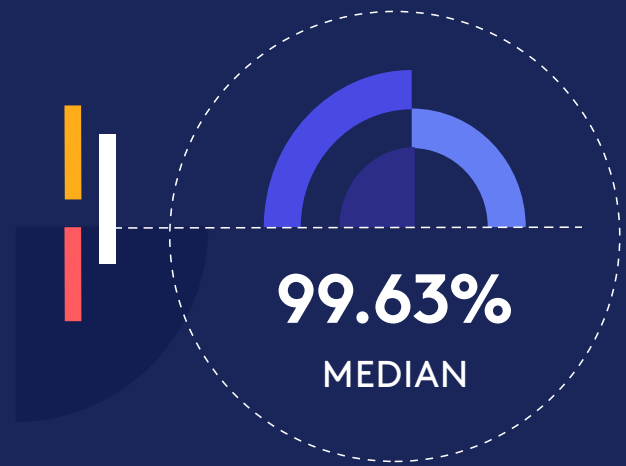
SLA → CRITICAL STABILITY

- Minimum threshold
- Drop everything and fix bugs

Mobile applications

A prime opportunity for mobile-first engineering organizations to measure and improve application stability

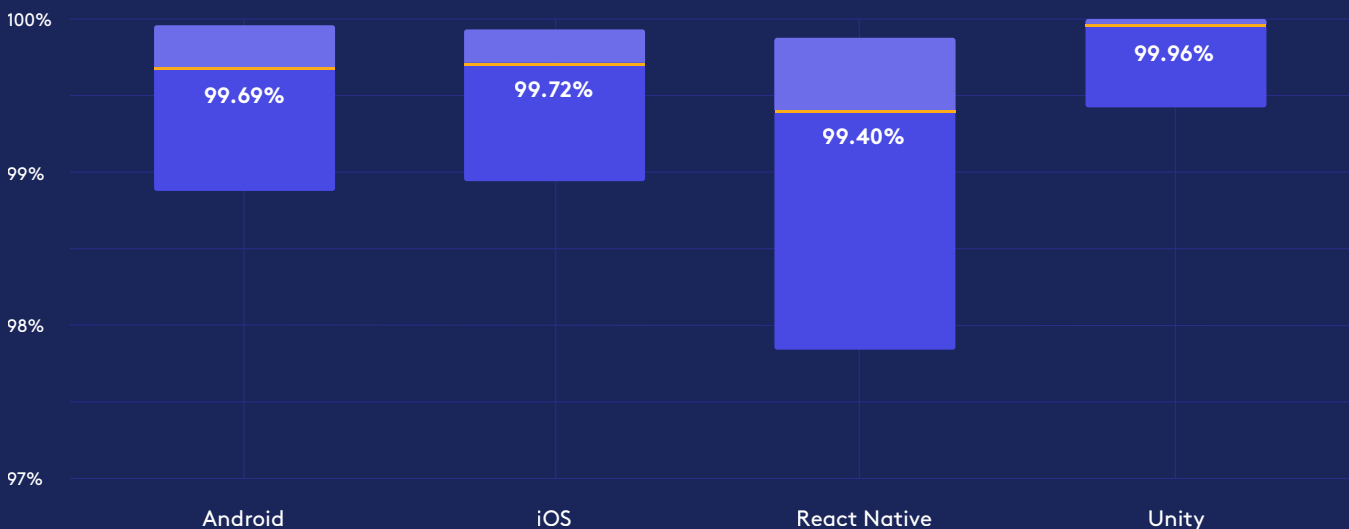
The data includes applications from several mobile development platforms, including Android, iOS, React Native, and Unity. Stability scores are negatively impacted by session-ending events, which include things like crashes as well as ANRs (Application Not Responding) in Android, React Native, and Unity applications and OOMs (Out of Memory) in iOS applications.

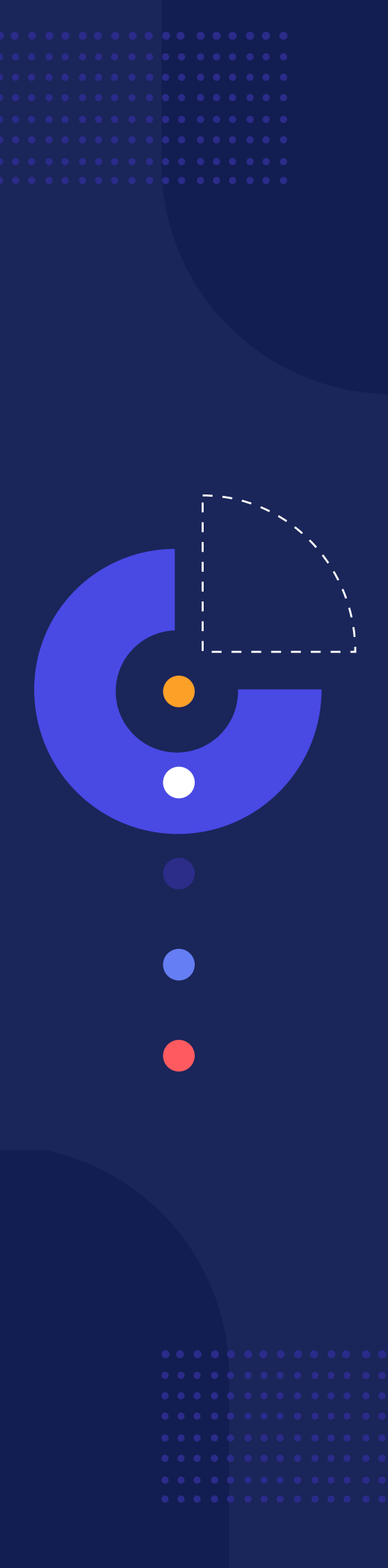


Almost one in every 250 customers could be having a completely broken experience with the application.

Compared to the “five nines” which are used to set goals for uptime and availability, a median stability of 99.63% presents a big opportunity for engineering organizations to invest in measuring and improving application stability and customer experience.

STABILITY SCORE BY TYPE OF MOBILE DEVELOPMENT PLATFORM





Android and iOS native applications tend to have a high median stability because there are very specialized developers working on these applications who have the expertise required to understand and address any stability issues effectively.

Android applications tend to have a slightly lower median stability compared to iOS applications because Android presents a much less constrained development environment. Increased fragmentation of Android devices makes it more difficult to test applications whereas iOS development teams only need to provide a stable experience on a limited number of devices that Apple releases every year.

More about [Android fragmentation](#)

React Native allows Engineering teams to develop for iOS and Android together, however, [writing code once to deploy everywhere does present some trade-offs](#). Oftentimes, React Native is used to open up mobile development to for example JavaScript engineers, who may not be accustomed to the challenges it presents. Expertise is also needed in Android and iOS to effectively investigate and fix bugs in the native layers of the code. This may explain the wider range and a slightly lower median stability.

More about [React Native](#)

Due to the sandboxed environment of **Unity** applications, bugs that crash the entire application are much less likely to occur. It is possible that errors within frames are disrupting the gaming experience, but they don't result in a full on crash, which may be why Unity applications boast the highest median stability and the narrowest range.

More about [Unity trends](#)

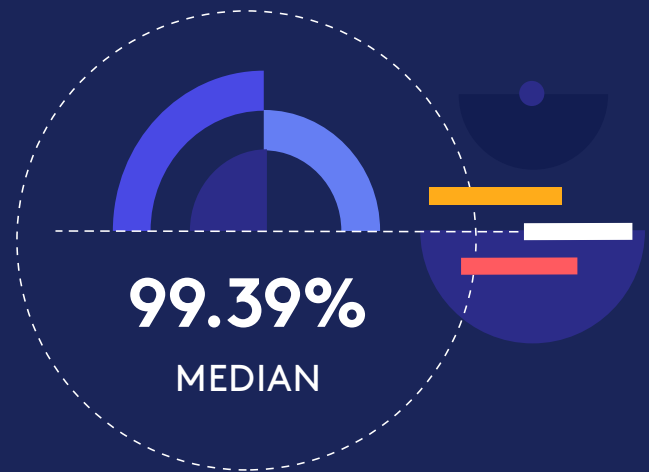
Web applications

Modern and opinionated web development frameworks may provide more resources to manage errors

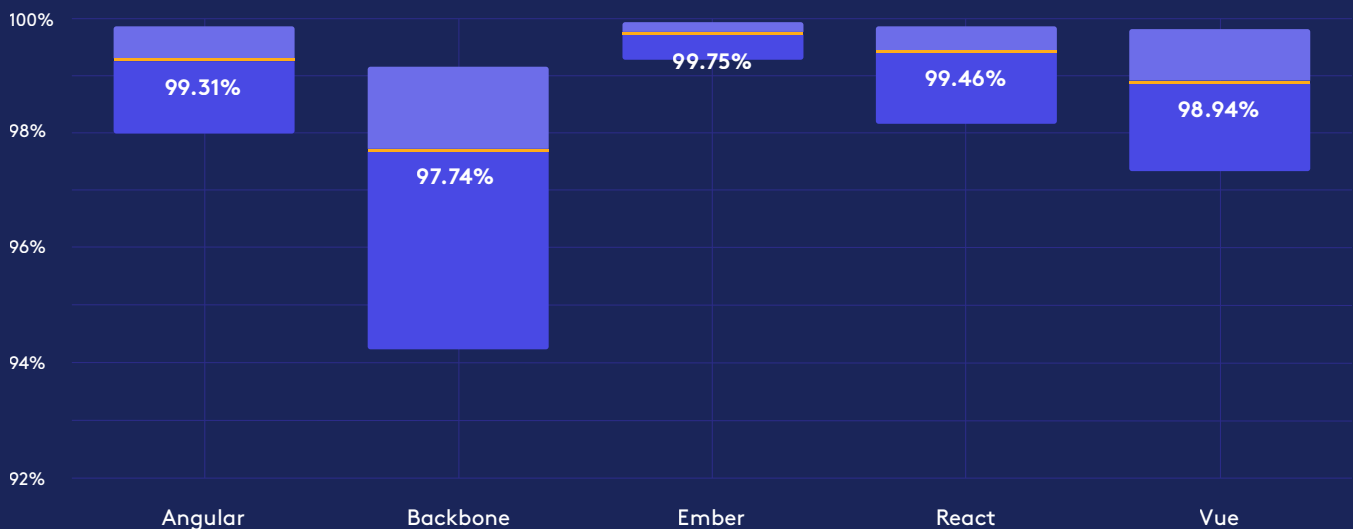
The data includes applications from several front end development platforms, including Angular, Backbone, Ember, React, and Vue. Causes of unhandled exceptions in web applications can include a bug

which prevents the entire page from rendering, an event handler bug which causes the user interaction to fail, an unhandled promise rejection warning, and more.

Web applications may have a lower median stability compared to mobile applications because monitoring and addressing client-side issues in JavaScript applications generally requires more effort than doing so in mobile applications. Mobile is a newer discipline and its development frameworks were built with a greater emphasis on the importance of managing errors from the get-go, whereas web is an older discipline that had to learn this over time.



STABILITY SCORE BY TYPE OF WEB DEVELOPMENT PLATFORM





Angular, Ember, React, and Vue are modern, opinionated JavaScript frameworks that are built with consideration for error handling. Angular and React were created and sponsored by development teams at Google and Facebook. Engineering organizations working with these platforms have access to the resources and documentation they need to investigate and fix errors that may affect application stability.

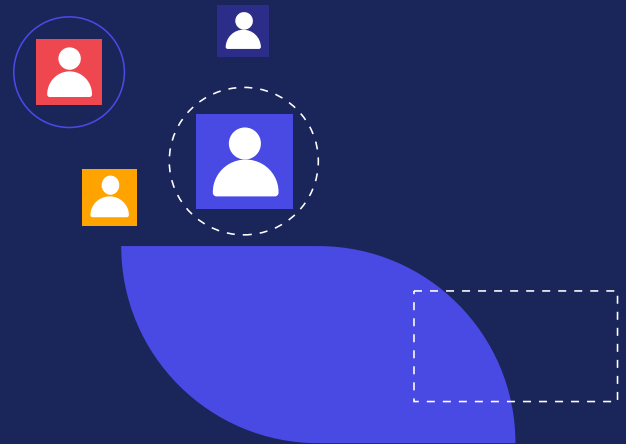
Backbone is an older and less opinionated web development framework. Development teams don't have access to the same coding guidelines, best practices, and considerations for error handling that the other more recent development frameworks offer, which may explain the lower median stability and wider range for Backbone applications.

The stability of each application is monitored across **browsers**, meaning customers experiencing errors in Internet Explorer, Google Chrome, Mozilla Firefox, and other browsers are included in the overall application stability score. Errors caused by **browser extensions** are also included; however, most engineering organizations don't spend resources to investigate and fix these errors since their code is not the culprit.

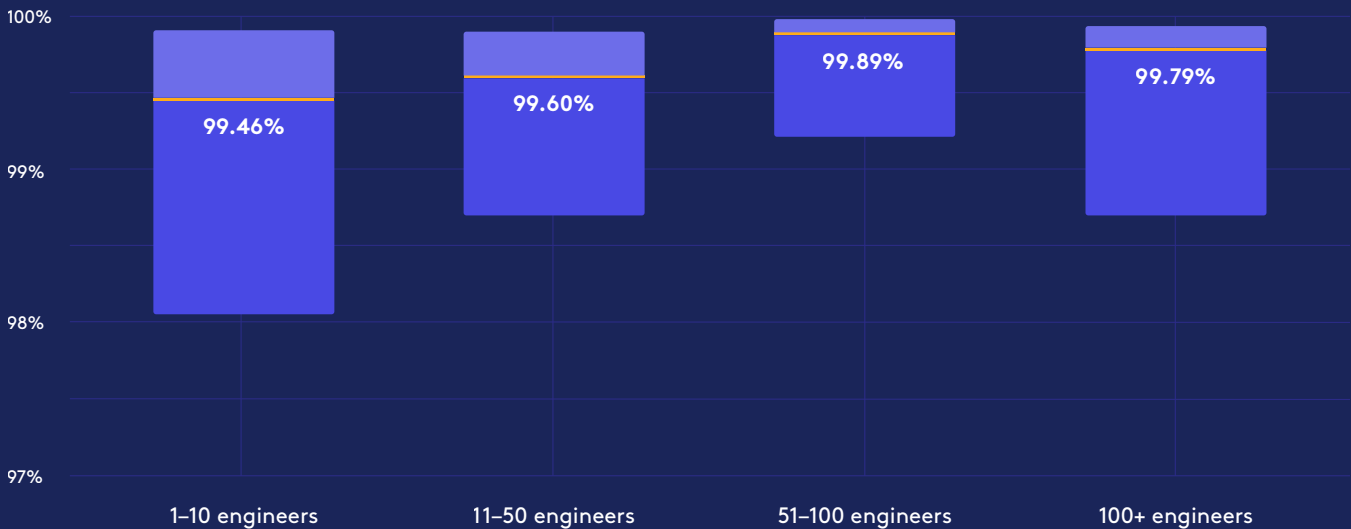
Engineering team sizes

Technical debt can make it more challenging for larger engineering organizations to maintain a higher stability

In order to understand if the size of an engineering team has an impact on application stability, we looked at the stability of applications supported by a few different engineering team sizes.



STABILITY SCORE BY SIZE OF ENGINEERING TEAM



The size of an engineering team tends to be linear to the age of the organization. Younger organizations prioritize **product-market fit** and need to release new features quickly. Maintaining a high application stability is less important than capturing market share and increasing competitive advantage. This may explain why the smaller the engineering team, the lower the median stability.

Larger engineering teams are usually working on more mature applications but standardizing on stability can become increasingly challenging. **Legacy code, compounding technical debt, and complexity of team structure** are just a few of the hurdles that can make maintaining a higher stability more difficult. This may explain why engineering organizations with more than 100 engineers have a lower median stability and a wider range.

How does application stability drive business outcomes?



Maximize customer engagement, retention, and business growth

You can no longer separate business metrics from the stability of your application. Increase application stability to deliver a better customer experience and drive engagement, loyalty, and business growth.



Protect revenue by fixing business critical errors

Know which problems in the applications are affecting business critical app functions or VIP customer segments. Then, get to the root cause of these problems quickly and roll out a fix to minimize business impact and protect revenue.



Reduce technical debt and accelerate innovation

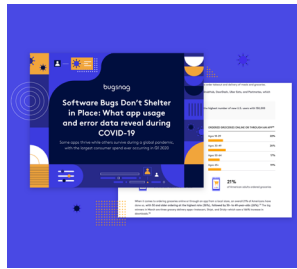
Align the Product and Engineering teams around when it's time to shift focus from developing features to addressing technical debt so you can accelerate the creation and maintenance of new code and solve customer problems in more innovative ways.



Increase competitive advantage by delivering new releases faster

Strike the right balance between software stability and roadmap velocity. Introduce new features before competitors do to increase competitive advantage and drive business growth.

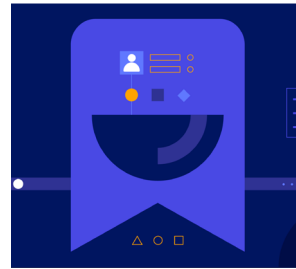
Additional resources



REPORT

What app usage and error data reveal during COVID-19

🕒 15 MIN



BLOG

Error alerting for your top customers

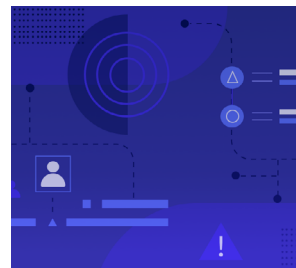
🕒 4 MIN



PODCAST

Measuring technical debt with Bugsnag CEO, James Smith

🕒 28 MIN



WEBINAR

Best practices for sustainably triaging bugs and defects

🕒 59 MIN

Methodology

- We used 30-day session-based stability scores from approximately 2,500 mobile and web applications.
- This is data from applications that are currently integrated with Bugsnag's error monitoring and stability management platform to measure and improve app stability and customer experience.



