



Unsnag Your Mobile Monitoring: Why Teams Are Switching to Sentry

Mobile error monitoring tools shouldn't flood your inbox or leave you numb to alerts—just surface what actually matters.





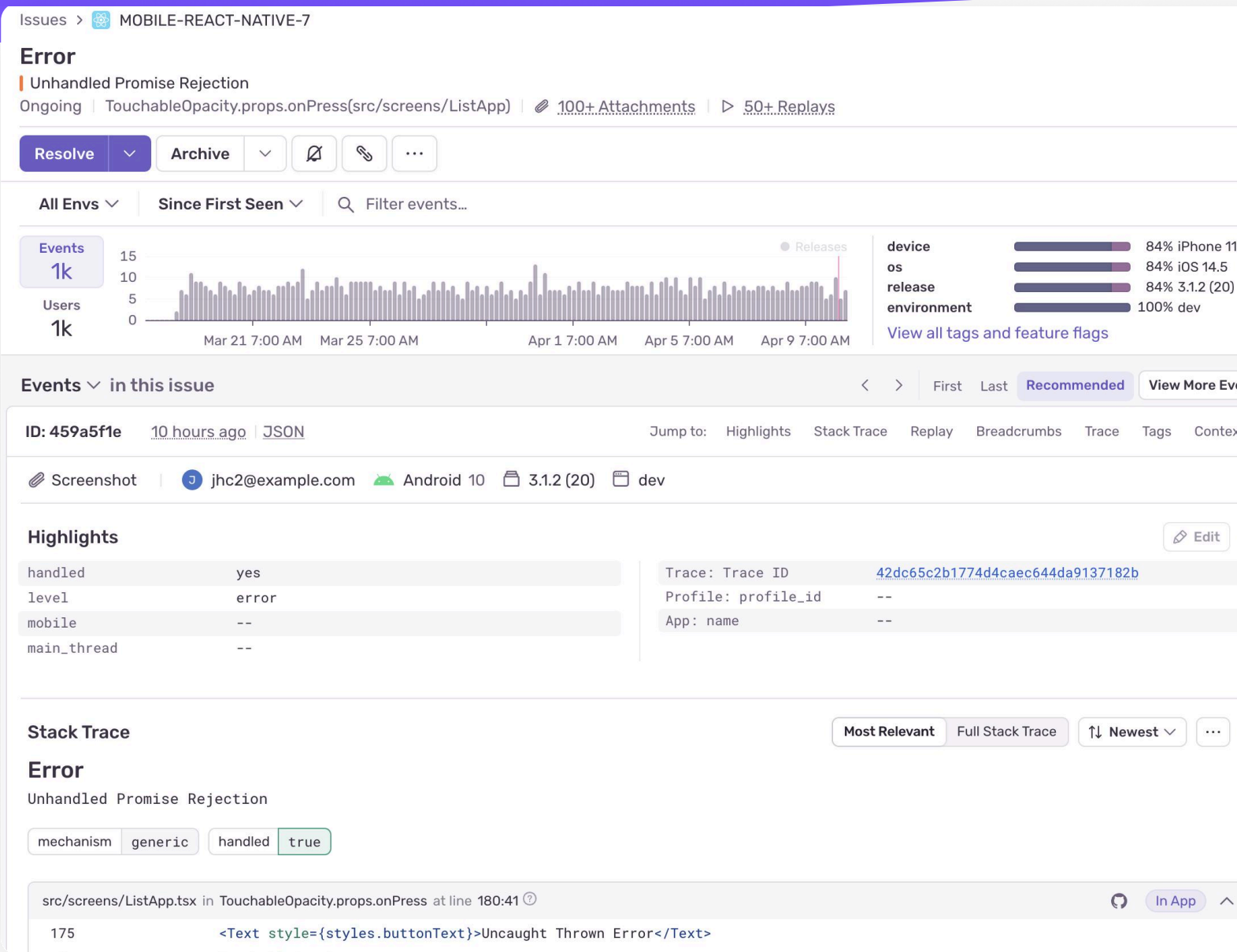
5 reasons mobile teams choose Sentry

Most mobile monitoring tools give you just enough info to know something’s wrong—but not enough to actually fix it. Crashes show up with little context, performance issues hide behind vague metrics, and you’re left piecing things together like a group project you didn’t sign up for.

Sentry gives you the full picture: errors, performance bottlenecks, and regressions, all with the context you actually need to debug like you meant to all along. Here are 5 reasons teams are making the switch (and wondering why they didn’t sooner):

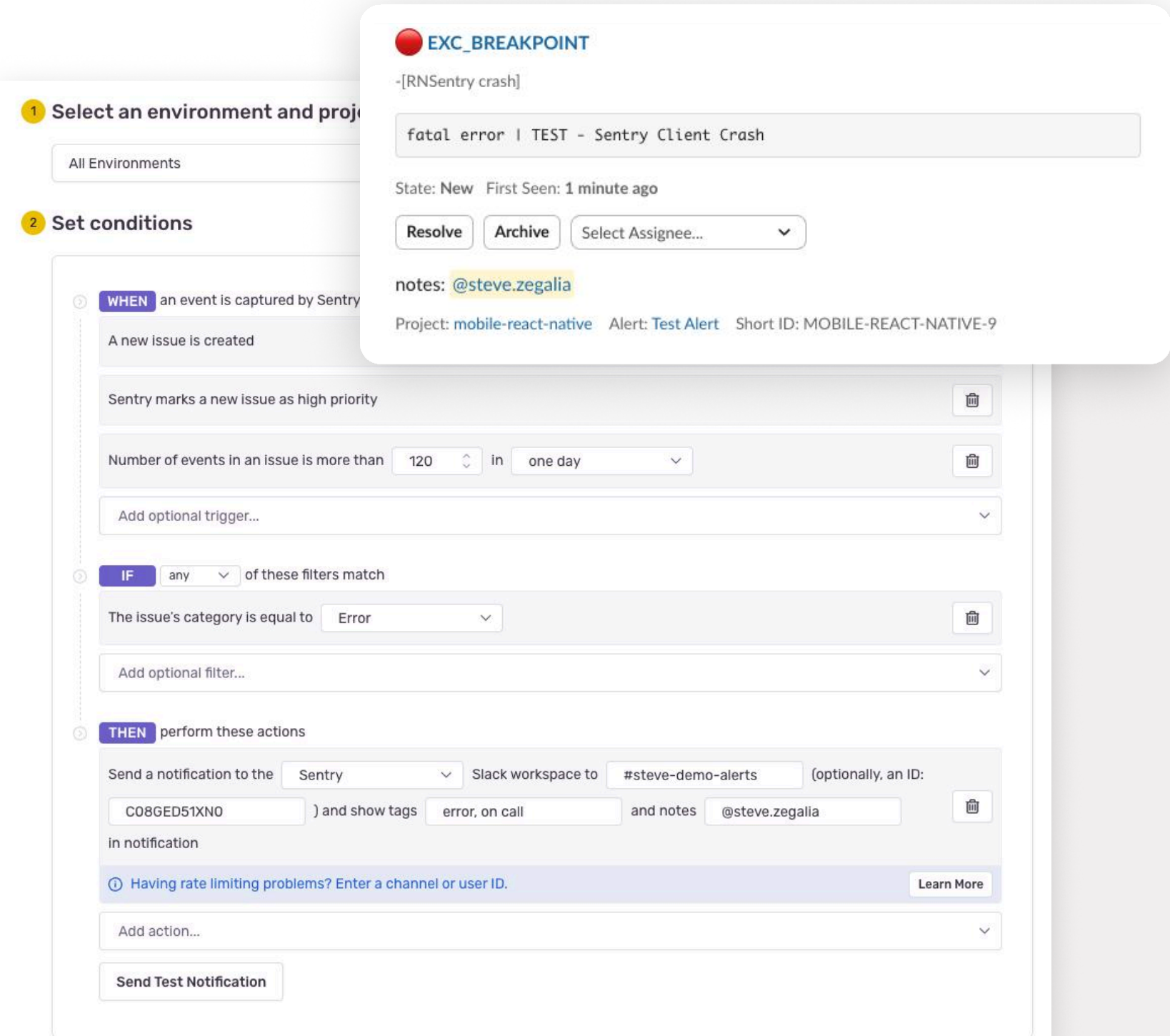
Reason #1: Get the right context on your errors

Sentry’s [issue context](#) gives you insight into the developer and commit responsible for the issue, with connected context from the stack trace, breadcrumbs, and mobile performance insights to help you quickly find the root cause and debug.



Reason #2: Error alerts without the noise

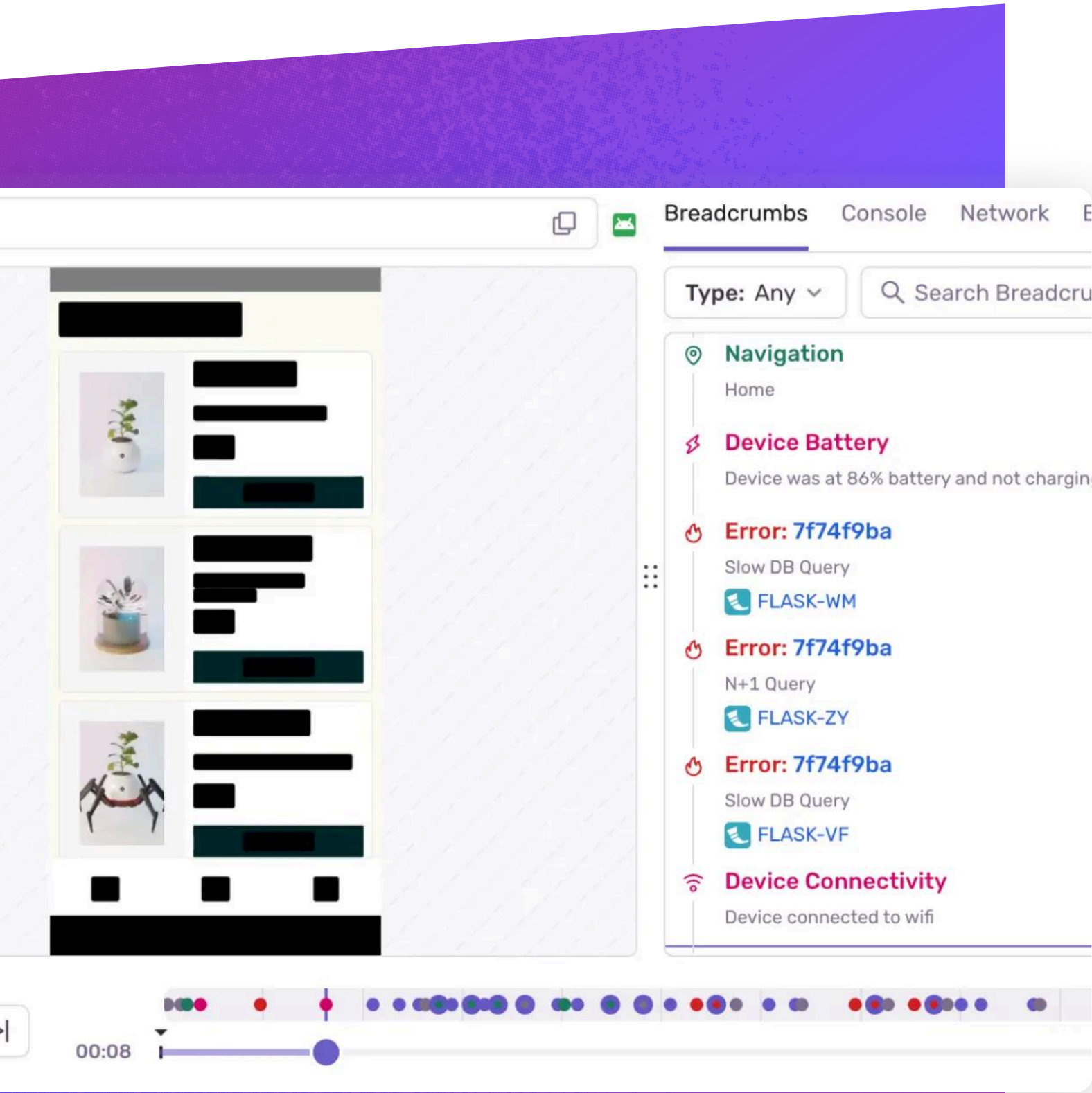
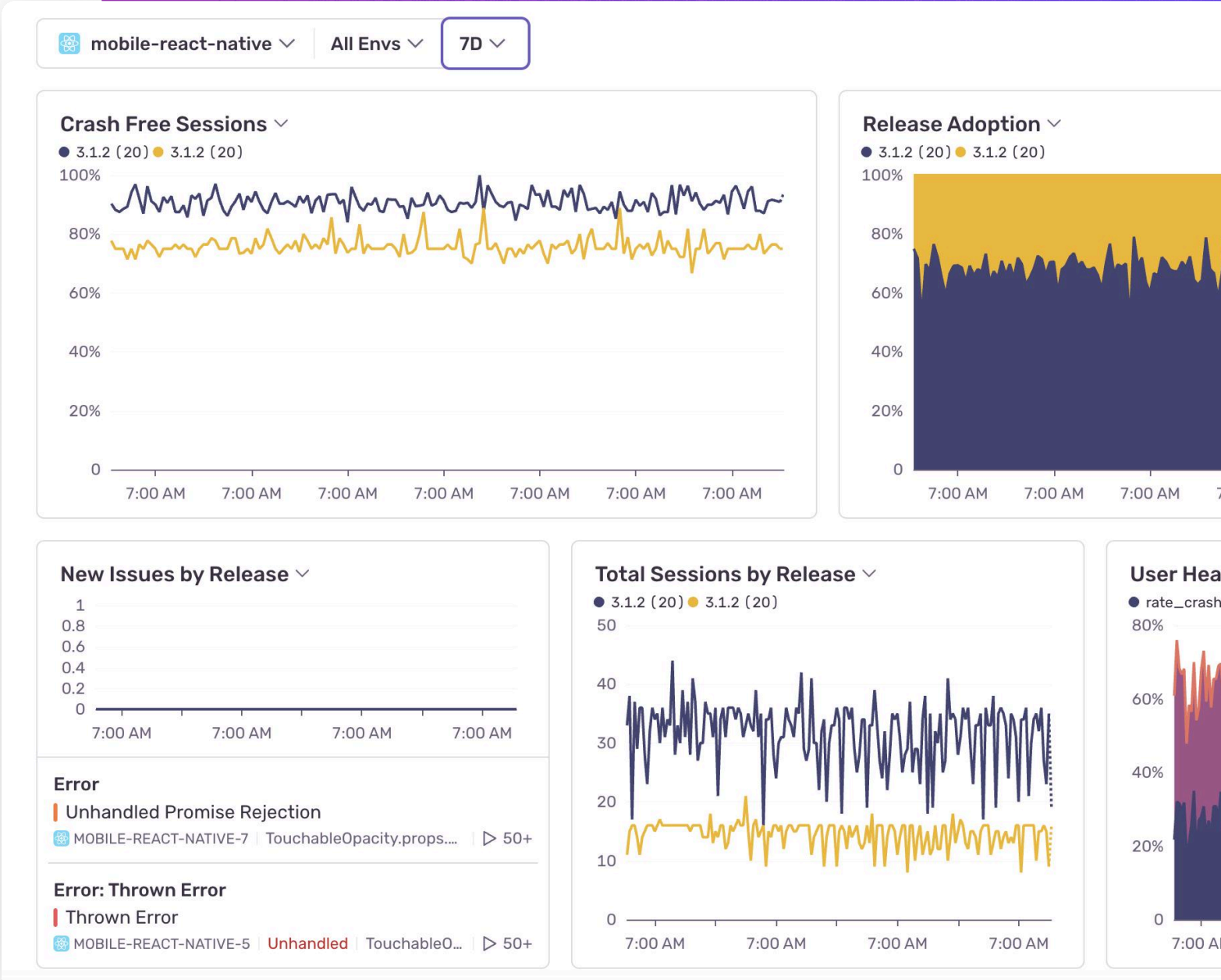
With Sentry, you can filter and prioritize [alerts](#) using tags, fine-tune notifications, and easily customize notification and error grouping rules so you don’t get fatigue from useless alerts and miss the issues you actually need to fix.





Reason #3: Queryable insights into app health and stability

Get instant insights into user sessions and [release health](#)—or dig deeper with [Discover](#) and use custom queries and filters to answer things like which parts of your code are causing users to abandon your app.



Reason #4: Watch reproductions of user sessions to repro issues faster

Use [Session Replay for Mobile](#) to get to the root cause of an issue faster by seeing exactly how users are impacted by errors, with replays connected to error events so you get context like device, battery, and network details so you don't have to guess when trying to reproduce the issue.



Traces > 77aab7834362

ui.load

Home | 1 Replay

564r@example.com Android 10 3.1.2 (20) dev

Search in trace

Legend: ui.react.mount 26%, db 14%, ui.react.update 12%, app.start.warm 7%

Timeline (TTID): 0.00ms, 1.00s, 2.00s, 3.00s, 4.00s, 5.00s

- Trace - 77aab7834362428cbcd92943339eea60
 - 17^ ui.load - Home
 - ui.load.initial_display - Home initial display (905.00ms)
 - ui.load.full_display - Time To Full Display
 - 2^ app.start.warm - Warm App Start
 - app.start.warm - Process Initialization (591.36ms)
 - app.start.warm - JS Bundle Execution Before React Root (80.00ms)
 - navigation.processing - Processing navigation to Home (292.94ms)
 - ui.react.mount - <Root> (309.18ms)
 - 1^ http.client - GET https://application-monitoring-flask-dot-sales-engineering-sf.apps (10.12ms)
 - 3^ http.server - products
 - 2^ /products.get_products - function
 - 4^ metric.timing - products.get_products.execution_time (2.21s)
 - get_products - db.connect (0.10ms)
 - db - SELECT * FROM products WHERE id IN (SELECT id from products, Autogrouped - db (297.23ms)
 - serialization - json (1.91s)
 - 1^ /get_iterator - function (0.27ms)
 - metric.timing - products.get_iterator.execution_time (1.50s)
 - No Instrumentation (486.08ms)
 - 2^ /ruby_cached_api_request - function
 - 1^ cache.get - ruby.api.cache:60 (1.01s)
 - db.redis - GET 'ruby.api.cache:60'
 - 1^ /api_request - function
 - http.client - GET https://application-monitoring-ruby-dot-sales-engi
 - ui.react.mount - <StyledProductCard>



Meraki



Talk to a Sentry expert to request a demo.

Join our Discord

Check out GitHub

Read our docs