Checkly SREBot

Daniel Paulus - VP Engineering

Checkly

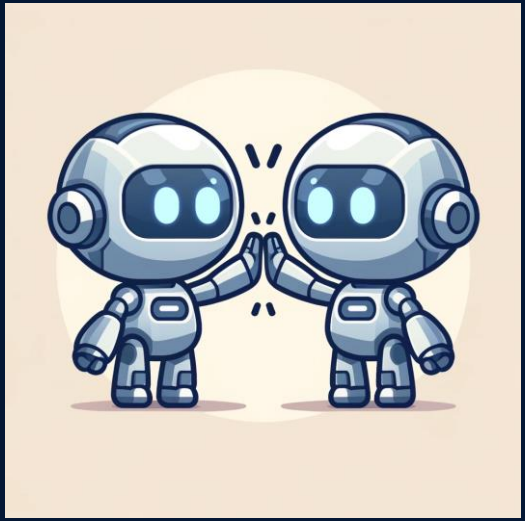
Empower developers to own and ensure application performance and reliability
- from pull request to post-mortem

Some thoughts on AI

"AI" is not a new thing. The basics have been around as early as in the 1940s.

Since then it has been developed and improved every decade especially powered by modern hardware.

Every iteration massively improved what these algorithms can do, and this will keep happening.
AI is not going away, but it will become a part of our lives more and more.

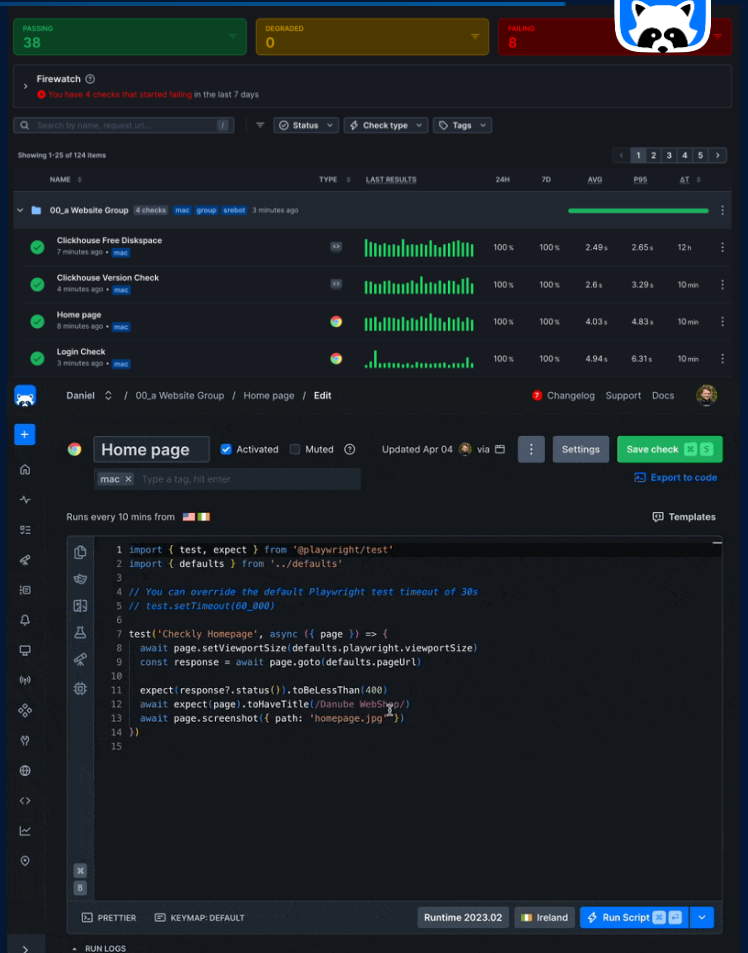


Quick word on what Checkly does

Checkly is a Playwright& API based monitoring SaaS.

Users write Playwright scripts for testing, and can use them as production monitors to run on a schedule and receive alerts if something fails around the globe.

They can also define API monitors, that will monitor all of their REST APIs, Websocket endpoints and much more.



How to approach an AI project in a start up?

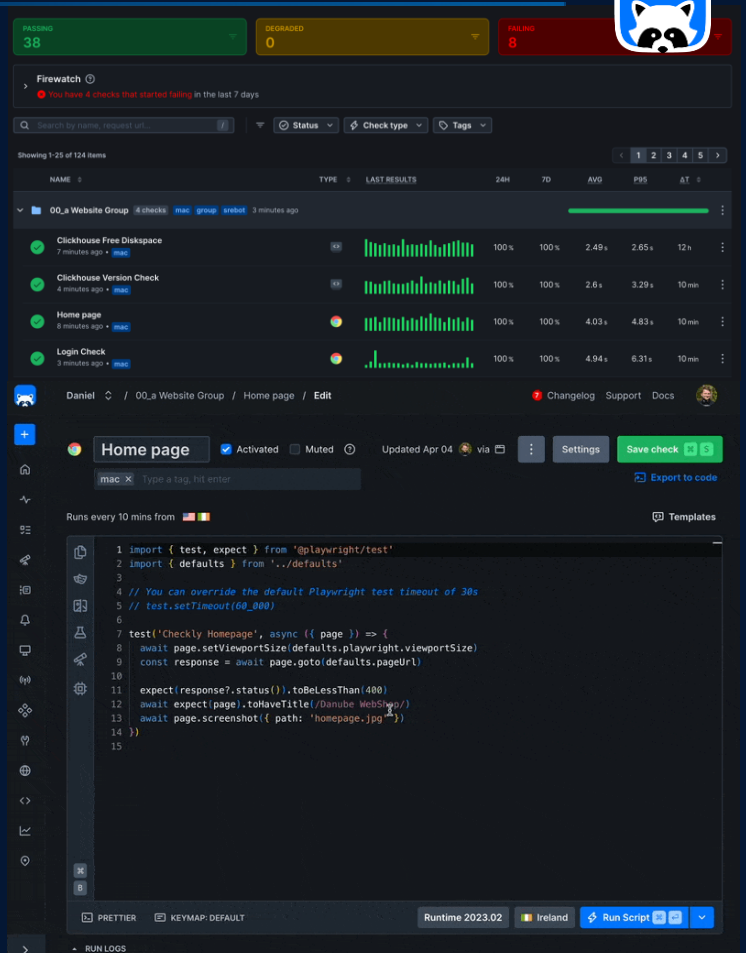
TL;DR

Like every other product development project :-)

We have endless ideas how to use AI, but which one should we go for?

=> We talked to 10 customers, running surveys to understand what problems they have

⇒ We picked one that seemed feasible to solve but also valuable enough for people to pay



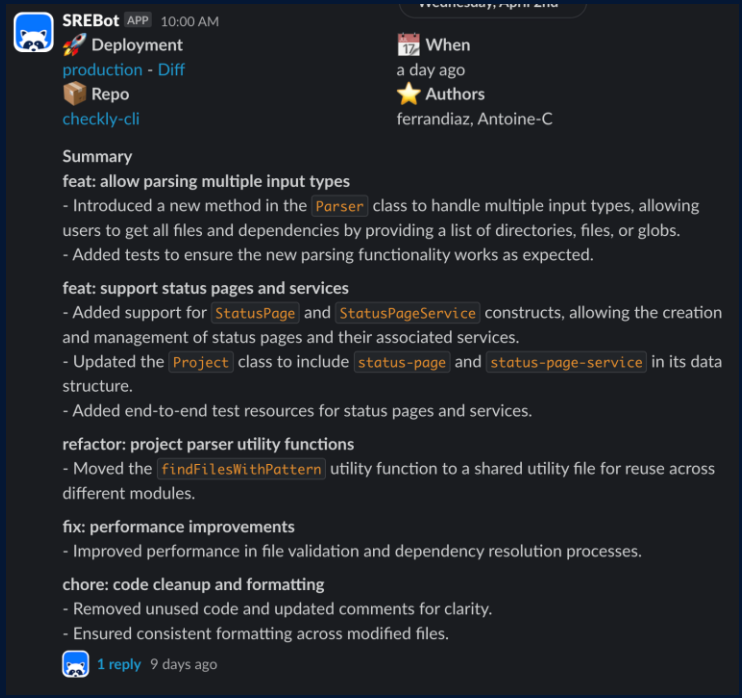
SREBot was born

It's your virtual Site Reliability Engineer to bring down MTTR and improve dev productivity while cutting engineering on call costs and stress by half.

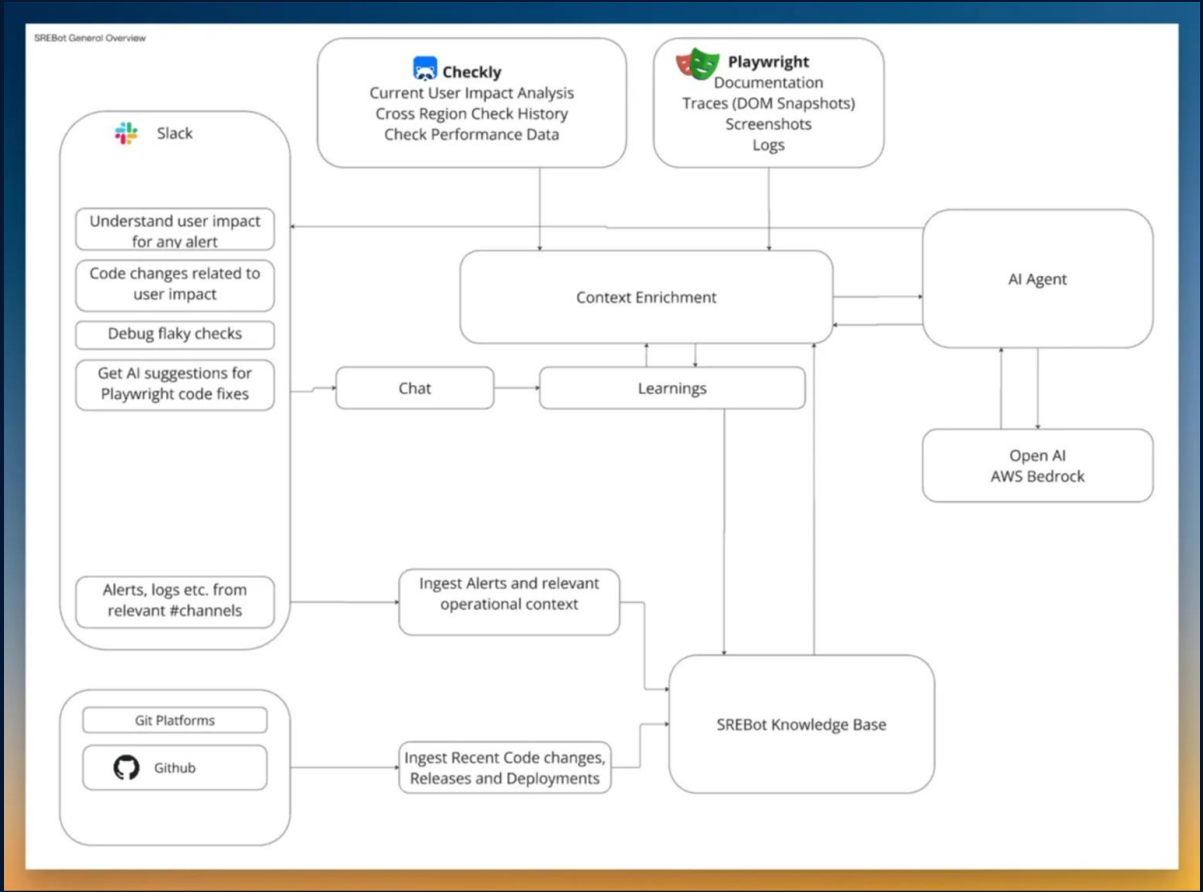
We talked to our biggest customers and for them this is a value prop they are willing to pay for.

We were closely working with Sales to secure design partners.

https://github.com/checkly/srebot



SREBot plan


SREBot General Overview
Checkly
Current User Impact Analysis
Cross Region Check History
Check Performance Data
Playwright
Documentation
Traces (DOM Snapshots)
Screenshots
Logs
Slack
Understand user impact for any alert
Code changes related to user impact
Debug flaky checks
Get AI suggestions for Playwright code fixes
Context Enrichment
AI Agent
Chat
Learnings
Open AI
AWS Bedrock
Alerts, logs etc. from relevant #channels
Ingest Alerts and relevant operational context
Git Platforms
Github
Ingest Recent Code changes, Releases and Deployments
SREBot Knowledge Base

Learnings

Tech considerations

- LLMs are really good at code summaries& text generation
- They are quite good at planning small steps
- They cannot magically figure out things, that a human wouldn’t be able to figure out either, you have to give context
- You still need to combine algorithmic and structured data with LLMs and Agents
- TypeScript works to build a system like this, but when you want to work with data, we often go back to Python

Learnings

Building prompts to give good results is the hardest part

- You tune a prompt using intuition, fix it for half the cases but break it for 10% you don't even know about ⇒ This is a data problem
- We tried langfuse for prompt observability
- LLM as a judge approaches
- Writing our own test harness

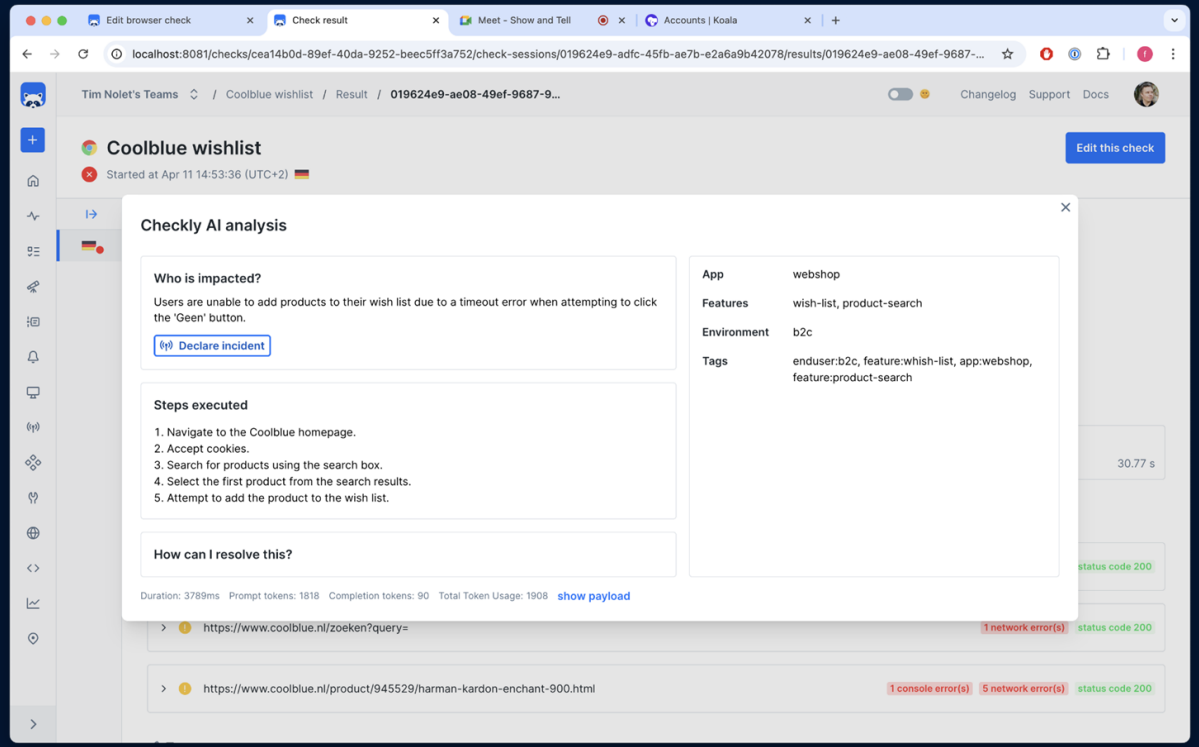
More Learnings& Conclusion

- It turned out that building a reliable SREBot like in the original idea was very hard
- Giving access to different observability systems was more than large customers were willing to do, the problem is real, but they did not trust us enough to give access
- We decided to leave source code changes out of the picture for now and focus on User Impact analysis. This is a real problem everyone has and we can solve it, because we own the data for it

⇒ Answer one question using AI, fast and with accuracy:

There is an incident or system event, what is the user impact?

Next Step: End User Impact Analysis



Any questions?

Thanks for listening :-)