



Audit report Sample

We're the market leader for technical due diligence of SaaS companies in Belgium and the Netherlands, servicing clients all over the world.

Investors appreciate our smooth process, sharp opinions, and top-notch reports.

Last updated: **April 1st, 2025**





IN THIS DOCUMENT, YOU'LL FIND:

- **An overview of our audit process** — what we audit for and why
- **The key questions we answer** as part of our audit
- **The lead time** of our audits
- **A bit more about us** — and how to get in touch if you'd like to gain insights into your engineering team or startup

Please note: some parts of this report have been blurred due to confidentiality reasons



Sample company audit report

2025-`{var_mm}`-`{dd}`

Audit outline

We (madewithlove) audited the Sample Company company and their application. We based the content of this report on interviews we conducted with the employees of Sample Company. Some of them also presented slides or a part of the codebase.

Our team that performed the audit:

- Brenden Thomas (CTO in residence)
- Emma Pieters (Software engineer & Technical writer)
- Geoffrey Robbins (Software engineer)

We interviewed:

- Sophie Pienaar (CEO)
- Lucas Degreef (CTO)
- Thomas Hofstetter (Product Manager)
- Olav Wellens (Frontend Engineer)
- Oumaima Shiraz (Data Science Analyst)
- Tomi Paternoster (CPO)
- Michael Hentilla

The information below is a summary of these interviews.

What we audit for

An audit usually focuses on one aspect, like the codebase's quality or the team's growth potential. Still, we touch on many topics during the interviews to get a complete picture of the company and its culture. We ask broad questions about the company culture, how communication is happening, how support works, etc. In a company, a problem rarely exists in one area only. It is usually a combination of multiple things that influence each other. By understanding this big picture, we can pinpoint the company's and product's challenges so that you have the correct information to take on these challenges in the future.



Every audit is performed by a **team of world-class experts** who look at 5 pillars
(critical areas)

Our report begins with a short reminder of them...



List of our experts who performed the audit

Short explanation of our approach to auditing



Sample company audit report

2025-{var_mm}-{dd}

Audit outline

We (madewithlove) audited the Sample Company company and their application. We based the content of this report on interviews we conducted with the employees of Sample Company. Some of them also presented slides or a part of the codebase.

Our team that performed the audit:

- Brenden Thomas (CTO in residence)
- Emma Pieters (Software engineer & Technical writer)
- Geoffrey Robbins (Software engineer)

We interviewed:

- Sophie Pienaar (CEO)
- Lucas Degreef (CTO)
- Thomas Hofstetter (Product Manager)
- Olav Wellens (Frontend Engineer)
- Oumaima Shiraz (Data Science Analyst)
- Tomi Paternoster (CPO)
- Michael Hentilla

The information below is a summary of these interviews.

What we audit for

An audit usually focuses on one aspect, like the codebase's quality or the team's growth potential. Still, we touch on many topics during the interviews to get a complete picture of the company and its culture. We ask broad questions about the company culture, how communication is happening, how support works, etc. In a company, a problem rarely exists in one area only. It is usually a combination of multiple things that influence each other. By understanding this big picture, we can pinpoint the company's and product's challenges so that you have the correct information to take on these challenges in the future.

List of people interviewed during the audit: employees are more likely to open up to us than to our clients **because their feedback remains anonymous.**



The introductory part of the report ends with an explanation of the **5 pillars** that we analyse during audits.



Team and leadership

People are the most valuable asset. We attempt to understand the company values and see if the team understands and aligns with them. For example, a high turnover rate might indicate a lack of shared values. We also look at how new employees join the team. Is there a straightforward onboarding process that helps them in their first days, weeks, and months? A new employee can signal how streamlined and documented the process and company are. Are teams siloed? Are the roles and responsibilities well defined?

Leadership significantly impacts the company and its processes. We try to understand how management plays a role in the company. Is there micromanagement? Does management have enough peers to challenge them? The team can use a long-term vision and roadmap to motivate and direct their decisions. Communication, verbal or written, is vital in any company.

Process

There are always processes, whether auditing a product or a service company. Are they clear to the teams? Is there transparency for the rest of the organisation? There is often a disconnect between the management and the people building the product. This disconnect could be a reason for features not being delivered, a high employee turnover rate, a low quality of work, etc.

Written communication

Documentation is the source of truth, both internally and externally. There are many types of documentation, such as technical documentation spanning architecture description, coding guidelines, setup guides for new hires, or the external documentation of a customer-facing API. Does the team effectively share knowledge? Are the processes clear to the entire team? Does the team effectively consider the customer's needs? The documentation helps to answer these questions.

Engineering

The product development team is usually the heart of the product and company. We examine the high-level architecture of the application or ecosystem and how all the components communicate. The specific issues are not necessarily important, but it is interesting to know if the development team is aware of them and if there is a plan to address them. Just like the company has a roadmap and vision of where it wants to go, the development team should also have this vision for its codebase.

Problem and solution

Having a clearly defined problem solved by a clearly defined solution is essential to success. It's natural for a product to evolve; however, it needs care and attention to prevent it from becoming too complex or ineffective. Are the team members aligned on the problem they are addressing? Are all existing components of the product considered necessary? Included in this is the prioritisation of new work. Involving customers is vital for building a well-defined product.

Bonus: We've written an article about **assessing seniority of devs and engineers** (click image to open).





Before we present the detailed findings, we provide an **Executive Summary** to address the key questions our client had about their team or product



This summary provides Yes/No answers to the key questions our clients care about most. These questions are not fixed; we define them individually with each client before starting the audit.



Executive summary

The team is of mixed skill and suffers from a lack of seniority. This is expected since the most senior engineer left the team in the last six months. Engineers feel a lack of support, especially from the CTO. That being said they have incredibly high trust in one another which is a key factor in high performance teams according to Google's research. The team is small so there are some key non-replaceable employees. This in and of itself is not a problem but there is little technical documentation. Team leaders should spend more time encouraging collaboration and providing feedback to engineers. This will help ensure that employees are engaged and able to shift to a quality first mindset.

The team is capable of learning from mistakes and improving processes. The team collaborates well at many levels but some processes prevent a full focus on delivering work together. Another example of this is the crisis response which is handled by one person primarily. Additionally, being process oriented will need to improve once the team needs to grow.

Technical documentation is overly missing. This makes it complicated to work on the existing system. The entire team should feel comfortable and empowered to maintain the documentation and expand it. The use of French in some documentation will be problematic if the team has non-French speaking employees.

The team is doing their best to deliver good code, but due to time constraints sacrifices have been made in the past. Specifically, the continuous integration and continuous delivery pipeline needs improvement to help the team deliver high quality software. Currently, there is a lot of legacy code which has been ignored. It needs to be integrated with the new solution and simplified. This will allow the team to manage the infrastructure and codebase in a more efficient way. One key example is that the billing system should be abstracted and replaced with a third solution as soon as possible.

The company needs to transition into a services platform. To do that, customer satisfaction will be key. Currently, there is very little formal product knowledge and so there are no end user interviews, product reviews, or a well structured and formatted roadmap. These will be key for the team to achieve their goals.

To answer the key questions directly:

1. Is the team ready to scale?

No. The CTO has abdicated the team management responsibilities. Although he is involved in hiring, the CTO mainly spends time programming instead of working on strategic or people management issues. This has affected the processes of the team. Recently, one of the Lead engineers left the team which has hurt team morale. The team will need to address these issues before they can begin to scale quickly.

2. Should the team refactor the existing product or start fresh?

In this case, the product is of good enough quality that the team can refactor it. This also makes sense because the product is not changing its audience or base technology. To make the effort



successful, the team should use the strangler pattern to slowly shift responsibility to an updated version of the framework.

3. Is the vision clear to the entire team?

Yes. The vision is clearly defined and the entire team is aligned behind it. The product manager has done an excellent job of documenting and communicating the vision.

4. Can the solution be easily copied?

No. There is a lot of custom code which can not be easily copied. A competitor could compete based on a simplified or subset of features. The main parts of the application which are complex are the recommendation engine which eventually should implement machine learning functionality. Currently, it is implemented as a rules engine which satisfies user needs.

5. Does the team share knowledge effectively?

No. Little collaboration occurs. Additionally, there is little to no technical documentation. Because of this, some key non-replaceable employees are solely responsible for parts of the application. This is problematic if they decide to leave the company in the future or are unavailable. The best way to resolve this is by documenting technical decisions and processes. Knowledge sharing via pair programming is another area that can create cross-training opportunities. When the processes and technical knowledge is shared more thoroughly throughout the team, Sample company will be prepared to scale.



Finally, the main part of the report —
Observations (*objective*) and Concerns (*subjective*)
for each of the 5 pillars that we audit

All observations fact-checked by the audited company to ensure validity





We discuss every pillar separately,
starting with
Pillar #1: Team and leadership.

We break down each area into smaller
observations. **They provide a clear picture of the
processes and relationships in the company,
which, very often, different team members
describe differently.** And that's how we can find
out if there are communication issues in the
team.



Detailed findings

Team and leadership

Observations

Psychological safety

The team trusts one another and works well together. Psychological safety is the number one trait of high performance teams according to Google's research titled Project Aristotle.¹

Team structure

The company is divided into two divisions: software and operations. The software team is composed of 7 people and the operations team consists of 3 people.

Roles and responsibilities

Each person understands the roles and responsibilities of the others. There is one exception, for the CTO, who has delegated most people management responsibilities and is spending more time coding than working on strategic initiatives.

Collaboration

There is very little collaboration. Pair programming occurs rarely, if at all. If an outage occurs, the team does collaborate to resolve the problem.

Feedback

Planned feedback occurs via 1-1 meetings between the Lead engineer and the other team members. This work has been delegated by the CTO who prefers to program. 1-1s generally occur monthly. There is little positive feedback received. An annual review exists to discuss salary and improvement opportunities with the CTO. Unplanned feedback rarely occurs.

Continuous learning

There is an allocated budget for each employee to attend conferences or buy books. Every team member is aware and has used the budget in the last year.

Retention

A lead engineer has recently left the company. There was a major disagreement between the CTO and the lead engineer which caused the lead engineer to find a new job. The disagreement concerned the refactor or rebuild decision for the product. The team has mixed feelings about the lead engineer leaving but it is unlikely that others will leave the company.

Unfulfilled roles

The team is currently lacking a specific expert in user experience design. Although the product manager has some experience, it is not enough for a product of this nature. Additionally, the team has not replaced the lead engineer who left. By doing so, more energy can be spent on mentoring the less senior engineers.

¹ <https://rework.withgoogle.com/print/guides/5721312655835136/>



Based on the Observations, we describe our Concerns — things that sometimes might not even seem that important, **but which are likely to escalate in the (near) future.**



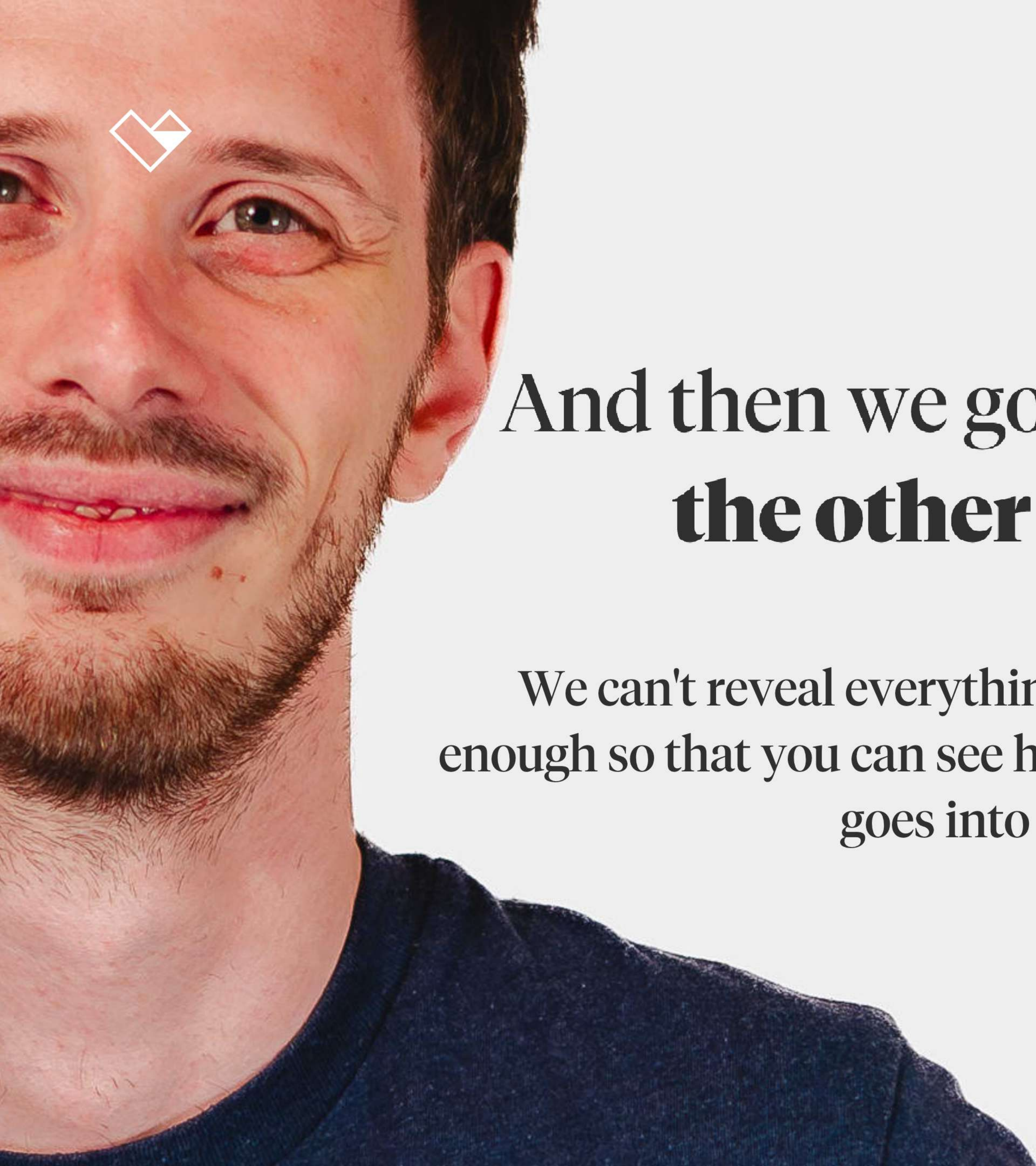
Concerns

- There are key non-replaceable employees**
Because the team does not collaborate or pair program often, there are some individuals who have unique and specific knowledge of functionality of the application. This is known as a bus factor.² This is natural for small teams and can be mitigated with good code review, collaboration, and documentation practices.
- Feedback is needed for employees to thrive**
Clear, frequent, and direct feedback is needed for employees to do their best work. Positive feedback is especially missing from the team. Curiously, this work has been delegated by the CTO to the lead engineer. Code reviews are an important part of giving and receiving feedback. The team might benefit from weekly or sprintly 1-1 sessions.
- Little to no collaboration occurs**
Engineers are largely working within silos so little collaboration occurs. Pair programming and other methods of collaboration are an excellent way to spread knowledge and cross-train the team. This will help mitigate the impact of key non-replaceable personnel.
- The team is missing expertise in user experience**
The role of user experience is currently handled by the product manager who has some background in this. However, for a product of this nature, it is important to have a dedicated user experience expert who can incorporate user interviews and quantitative data to create a seamless user experience.



Bonus: We've written an article about **why startups fail** — based on auditing 50+ companies (click image to open).

² https://en.wikipedia.org/wiki/Bus_factor



And then we go on to describe
the other 4 pillars.

We can't reveal everything, but we'll show you just
enough so that you can see how much work and expertise
goes into the report



Pillar #2: Process

An example of the Observations we made about this client's hiring process, as well as our Concerns about their deployment process.



Concerns

There are key non-replaceable employees

Because the team does not collaborate or pair program often, there are some individuals who have unique and specific knowledge of functionality of the application. This is known as a bus factor.² This is natural for small teams and can be mitigated with good code review, collaboration, and documentation practices.

Feedback is needed for employees to thrive

Clear, frequent, and direct feedback is needed for employees to do their best work. Positive feedback is especially missing from the team. Curiously, this work has been delegated by the CTO to the lead engineer. Code reviews are an important part of giving and receiving feedback. The team might benefit from weekly or sprintly 1-1 sessions.

Little to no collaboration occurs

Engineers are largely working within silos so little collaboration occurs. Pair programming and other methods of collaboration are an excellent way to spread knowledge and cross-train the team. This will help mitigate the impact of key non-replaceable personnel.

The team is missing expertise in user experience

The role of user experience is currently handled by the product manager who has some background in this. However, for a product of this nature, it is important to have a dedicated user experience expert who can incorporate user interviews and quantitative data to create a seamless user experience.

Process

Observations

Hiring

An applicant goes through 3 phases during hiring. First they have a screening call with the COO and CTO. Next, they perform a technical task to evaluate their skills. Finally, they meet with the CTO and Lead engineer to discuss the results of their task. An applicant tracking system is used. For the most recent job posting, 35 applications were received. Adverts were posted on the company website only.

Onboarding

New employees are well supported by the engineering team and have constant mentorship and attention. Learning the technology and product can be difficult due to the lack of documentation.

New hires

The team does a good job of using pairs and subteams in Jira. Context is provided by wikis and Confluence documentation but, at times, it is too vague. The product owner creates detailed diagrams, decision trees, and following messages.

² https://en.wikipedia.org/wiki/Bus_factor



Concerns

There is no formal outage response or on-call rotation for engineers.

CDPR

There have been no CDPR requests to date.

Concerns

Deployments can occur more frequently

The team currently deploys roughly once per month. Deployments are performed manually via bash scripting. A simple CI pipeline does exist but only runs linter tools rather also including automated tests and continuous deployment actions. By investing in a complete CI/CD pipeline, the team will be able to bring changes to the staging and production environments within 15 minutes as opposed to 1 hour.

Quality assurance process is not always followed

There exists a quality assurance step before work is delivered to the product team. At times, the product team will begin testing before quality assurance has completed their analysis. This causes duplicate bugs to be filed which causes extra administrative work. The policy should be agreed upon and followed to prevent extra work.

No on-call rotation exists

Currently, if there are infrastructure issues, only one engineer can resolve them during off hours. As the company grows, an outage response and on-call process will need to be implemented.

Tasks are often underspecified

What is obvious to a stakeholder may not be obvious to the person implementing the feature. Lack of context and required details forces the team to go back and forth clarifying the task, which reduces precious time to be wasted.

Deadlines are poorly established and communicated

Business wants strict deadlines without consideration for the technical team's capacity. It is impossible for the technical teams to deliver solutions because of unrealistic technical debt. Estimating work usually helps resolve this situation by surfacing mismatches in the amount of work required versus a deadline, but estimates created by engineers are often unreliable because of the technical debt.


Technology decisions are made without involving the engineering team

Because of the complexity of existing systems, the engineering team should be involved in the analysis and design of system changes. There have been instances where decisions were made by product or business people without involving the party responsible for executing the plan, the engineering team. As a result, there can be no guarantee the decisions align with the engineering team's capabilities.



Pillar #3: Written communication

We made Observations about the way this team communicates (in writing) and expressed Concerns about the lack of documentation which new hires could learn from (or long-term employees could refer to).



Code review is not practiced on some projects.
Code review is a process of reviewing the proposed change by another engineer before the change is included in the project. It is practiced in most, but not all engineering teams. Code review, similar to pair programming, helps with knowledge sharing between team members and increases the quality of new code."

Written communication

Observations

Communication tools

The team uses Slack and the Atlassian suite to communicate. Confluence is used to document non-technical items and technical documentation is kept in the repository.

Language
English is primarily used for technical documentation and communication since the Bulgarian team does not speak French. There are examples of French being used in pull requests. Some Human Resources documents are in French.

Non-technical documentation
Documentation covering processes such as requesting time off are documented.

Technical documentation
Product documentation covering technical topics exists but it is outdated and incomplete. The team is aware that more technical documentation is needed but engineers do not provide this as part of their work. Quality Assurance documentation is also available in Confluence. Infrastructure design is documented but in French.

API documentation
Automatically generated API documentation exists since the team uses Swagger.

Customer facing documentation
There is a customer facing help center which provides solutions for end users. It is available in French and English. There are no metrics tracking specific pages.

Concerns

Operational documentation is lacking

There is little operational documentation. This leaves new hires with many general questions they have to seek an answer for themselves. Additionally, long-term employees cannot refer back to specific documentation easily.

Technical documentation is lacking
Technical documentation kept as close to the code as possible is important for newcomers to quickly become effective. Documentation can and should include topics like architectural diagrams, technical notes, pull requests, and a glossary of domain terminology.

madewithlove

page 10 of 18



Pillar #4: Engineering

We've saved (or managed) dozens of tech companies and we know which infrastructure, codebase, and lack of best practices will eventually lead to a failure of a business or product.



The quality of the technical documentation is not consistent. There is a discrepancy between technical documentation from repository to repository. At times, the documentation is robust with detailed instructions and references. At other times, readers are left with a lot of environment variables. Each repository should cover at a minimum brief description, installation instructions, a link to the roadmap, and support details.

Future hires won't be able to read documentation. Almost all of the technical documentation is in English. However, some parts are in German. This is problematic for non-German speaking engineers. Additionally, the entire team should be engaged in documentation writing, and this is not possible if some team members cannot understand the language used.

Engineering

Observations

Infrastructure

AWS is used to host the staging and validation environments. The production environment is isolated in an entirely separate account. CloudFormation is used for orchestration and automatic load balancing. Terraforming allows for Infrastructure as Code.

Environments. Most system components have staging and production environments, some also have an additional development environment.

Monitoring. AWS CloudWatch is used to monitor the production infrastructure. It sends notifications to email and Slack. There is a television in the office which displays CloudWatch charts. CloudWatch is not fully configured to monitor CloudFormation. There is no specific performance monitoring. Other infrastructure is monitored using New Relic.

Exception logging. The team has employed Sentry to log exceptions which is also configured to send notifications to Slack.

Data storage. Two types of data storage are used. AWS RDS is used for relational data, there is a Redis instance used for key-value data.

Data backups. Backups are configured and automated by AWS. The backups were recently tested when an engineer accidentally deleted some production data. The team believes that this system can be improved since the backups are not stored on S3 and are not frequent enough.



Code legibility. The code is sometimes difficult to read due to paradigms that introduce complexity. Additionally, there is no consistent code style or naming system enforced via continuous integration.

Comments and merge requests. No templates are used. Pull request descriptions vary in quality, with some lacking context such as an overview of all changes. Frequently, commit messages do not describe the atomic change in a clear way.

Changelog. The engineering team does not create a changelog for use by the product team.

Production access. Only senior engineers have production access.

Developer experience. The entire application is dockerized including a docker-compose.yml for orchestration. The application proved difficult to set up and run locally due to a lack of documentation and automation. Developers generally enjoy working with the codebase.

Concerns

Backups aren't automatically tested

The databases are backed up automatically but are not automatically tested. Approximately one month ago, engineers inadvertently tested the production backup by restoring the data locally to troubleshoot a customer issue. Regular and automated backups are only valuable if they are also tested.

Quality assurance testing should be improved. On most projects, there is no testing in place to enforce an agreed upon code style. Static analysis is only used on shipping API based on Spryker with PHPStan.

There is little test automation. Automated tests are essential to iterate rapidly in an existing codebase while maintaining quality. Currently, there are few tests only some components like the Laravel based API of the old shop and the Spryker integration of the new shop. There are no UI component or end-to-end front-end tests. It is a best practice to have functional tests to prevent regression issues as well as end-to-end tests to ensure critical business paths keep functioning.

Some systems lack continuous deployment. Older and the old shop do not have a continuous deployment system configured. As a result, deployment is a manual process that is potentially prone to errors.

Little to no refactoring has occurred. While building new features or making changes to existing features, little to no refactoring of surrounding code occurs. Code is added to follow the existing paradigms without looking for continuous improvements. Small incremental improvements to code quality and deliberate refactoring in areas that are frequently changed would improve the overall quality of the code.



Pillar #5: Problem & Solution

Finally, we check if the product vision is well defined, if the end-user's needs are understood, and, most importantly:
if the product is ready to scale or will it eventually fail?



Code legibility is problematic

Legacy is an old system. As such, some of the code contained within does not follow the old design is legible. Consistent code style enforced by automated testing and code conventions can help keep the code base more legible and maintainable.

Pull request quality can improve

Whenever pull requests are used, care should be put into supplying a clear description of the change proposed (additions, removals, fixes, deprecations, etc.). Pull requests should be linked to related issues which usually provide further context for the reviewers.

Product information is sourced from several systems

Product information management systems are not used. Because there is no single source of truth, including product data, conflicts arise in pricing and stock levels. This issue was identified but never fully implemented.

Problem and solution

Observations

Product vision

The product vision is clearly defined and the entire team is aligned behind it.

Product team

Some of the features, temporary custom code has not been removed from the code base. There is never allocated time to clean up code after past one time events.

Feature team

Features are sourced from multiple internal departments. They often come with conflicting priorities.

End-user research

Direct customer research is largely missing from the product shaping work.

Prototyping

Visual changes have been tested in the past with end users. When the new design is implemented, an initial prototype was developed to test framework functionality. Technical proof of concept is tested in isolation.

Validation

No product validation occurs. Software is built based on unvalidated requests. When delivered, departments sometimes request further modifications that are not described during the initial context. Feature impact is not reviewed.

Refactoring

No formal product roadmap exists. Software built by the team is treated as a collection of features rather than a product.



Task prioritization

No formal product task prioritization occurs. Instead, the team is driven by marketing and immediate business deadlines. Previously tasks were prioritized by the CEO and CTO. The team also had a Product Owner who would help manage task priority. Often engineers are working on several projects or sub-tasks at a time so work is additionally unfocused.

Metrics

There are no specific product metrics. Basic user activity is tracked using Google Analytics and reviewed by the marketing team. No custom events are tracked.

Concerns

End user interviews are not performed

Whoever takes on responsibility to manage the product will need to have regular, formal interviews with end users, whether they are internal (in the case of APIs) or external (in the case of a webshop).

The backlog is very large

Backlogs can almost always be purged once important ideas will resurface. This serves as added pressure and a distraction to more important work. Focus on providing immediate value now and have a handful of items ready once the current work is completed.

Teams are often interrupted by changing priorities

Even though the scrum process was applied, constant interruptions with unplanned feature requests resulted in no sprint to be completed. Not only this has a negative impact on the business, but also negatively affects the team morale over an extended period of time.

There is no product roadmap

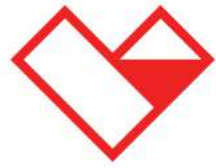
There is no vision which combines product needs with necessary technological upgrades. A roadmap is essential to communicating to engineering and stakeholders. To create a truly great roadmap, define issues based on outcomes or problems. This allows for solution ideation to occur by the technical team.

Task prioritization is not done by a formal method

Highest Product Owner's Opinion (HPOO) is currently used to determine the order of tasks. While it has worked thus far, the team should consider implementing a more formal method such as Weighted Shortest Job First (WSJF).

Time is not taken to cleanup issues after shipping

However fast paced the industry may be, removing features that will never be used again in the codebase stands in the way of keeping the project clean and flexible for future modifications. Time should be regularly allocated to remove unused features or clean up the codebase once, otherwise the surface for potential bugs increases. This is usually solved by creating a platform as a service for backend and essential systems which are then consumed by thousands of short lived solutions.



Lead time of our audits

We don't expect a thorough preparation from your side.

How much time do we need to perform our audits? Our ambition is to finish it within two weeks, but it also depends on your availability. All our meetings happen via video call. This is what we do:

- Validate key questions and focus points with investor (*)
- Have a baseline interview (2 hours)
- Conduct up to 7 additional interviews (7x 1 hour)
- In depth code review
- Debrief startup for fact checking (1 hour)
- Debrief investor (1 hour) (*)

() We don't do these if it's an internal audit and there is no investor involved*



CONTACT

Ready for an audit?

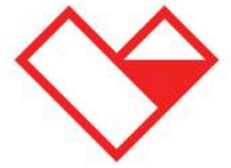
With our technical due diligence service, our staff engineers and experienced CTOs will technically de-risk your next funding round. In only 2 weeks.

Book a no-commitment call

(Secret: There are a few more slides left!)

Follow us around





WHAT WE DO

We step in when others step out. We enhance software, transform legacy code, derisk investments and strengthen tech teams. Fearlessly honest, highly experienced, and obsessed with quality, we bridge the gap between technical and non-technical teams. When the stakes are high, we make the difference.



WHO WE ARE

madewithlove, a world-class team of engineering experts

Our members are hand-picked masters of their crafts. We operate world-wide. Diverse cultures, interests, and backgrounds are a part of our company values, while physical borders are not.

You can find us in coworking spaces or home offices in Belgium, the UK, France, The Netherlands, Nigeria, Germany, South Africa, and Poland.



(click image to meet our team members)



WE WORK WITH INVESTORS ON A DAILY BASIS



FORTINO

THE
FAKTORY



Smartfin





madewithlove in numbers

100+

products shipped
worldwide

150+

startups audited
(and saved)

941+

amount of interviews
conducted (and counting)



CONTACT

Get in touch

Our technical experts are ready when you are

Andreas (our CEO) will jump on a **free, no-commitment call** with you, to find out why you need an audit, and explain what the best next steps would be — **in your specific case.**

Book a no-commitment call

Follow us around



@ audits@madewithlove.com

Andreas Creten, CEO