KOMODO

A COMPREHENSIVE 10-POINT SOFTWARE AUDIT

A complete framework for assessing the quality, maintainability and supportability of a software engineering project.

— WORK BOOK
SOFTWARE AUDIT

WHAT'S IN THIS WORKBOOK

CONTENTS

Estimated reading time: 20 mins

01 — RATE MY TECH

RATE MY TECH – KOMODO STYLE

As a technology decision-maker, you're facing a lot of unique challenges. Alongside the day-to-day responsibilities of your role, all eyes are on you to make sure your product roadmap is delivered on time AND on budget. That's a lot of pressure.

We realise that critically assessing your current technology can be a hugely time-consuming process. We also know a successful technology audit can be the difference between your product failing or succeeding long into the future.

Whether you're relatively new to your responsibilities or a veteran in the C-Suite, performing a tech audit can be overwhelming. That's exactly why we've produced this comprehensive 10-point technology audit for you.

01 — RATE MY TECH

Using our 18+ years of experience in software development, we've created and fine-tuned a project scoring criteria that really works for us. We use it to rationalise our understanding of a codebase and a project based on:

– How easy it is to work with and understand.

– How secure and compliant it is overall.

– How well it performs.

– How well it's been maintained over its current lifecycle and previous ones.

– How challenging it will be to add new features and functionality.

– Whether there are any project roadmap concerns.

– What risks may be faced supporting and maintaining your codebase – modifying, refactoring and expanding the feature set.

At KOMODO we believe in writing the best code we possibly can. While it's a treat to start a fresh codebase with best practice baked in from day one, that's not always possible – business needs must come first.

02 — WHY AUDIT

WHY EVEN CONDUCT A TECHNOLOGY AUDIT?

We can tell you an audit is best practice and all the top CTOs are doing it (agency spiel, etc.). But what are the real benefits?

Auditing your software helps you understand potential security risks. Gartner estimates only 24% of companies follow security best practices; cyber attacks resulting from poor security were projected to cost over $6 trillion in 2021. We don't fancy chipping into that pot.

According to CIO, a technological audit improves the innovation process by creating a risk profile for current and future projects. On completion, you can focus on the company's experience with those technologies and where it stands in the market.

We've been through a lot of technical audits and seen first-hand the impact they can have on software projects in the long run. They improve productivity, put less stress on your team, and inform better strategic planning and smarter decision-making – all while reducing costs and speeding up delivery.

At KOMODO, we aim to understand a project quickly and effectively. We use this detailed audit to evaluate the risks, costs and benefits of working on a project, codebase and technology set. We hope it helps you understand the strengths and weaknesses in your current technology – and steer you to success.

What to expect

– 10 actionable assessment criteria to gauge the condition, quality and consistency of your software and code.

– A practical rating system to benchmark and understand your unique position.

– Insider knowledge and insight into the auditing process, and what to look out for.

– A straightforward checklist and scorecard to track progress and create a targeted action plan.

03 — HOW TO APPLY

HOW TO APPLY THE AUDIT

Our framework is underpinned by a 0-to-3 point scale for each of the 10 individual criteria.

Our scoring system

Each category of the assessment scores up to 3 points, making the maximum score 30 points.

The overall score is then rated using RAG (Red / Amber / Green). This gives us a clear understanding of the project, which informs how we want to proceed with a client's project.

03 — HOW TO APPLY

RAG rating

The overall score is rated Red, Amber or Green – giving a clear read on the state of a project at a glance.

RED · 0–9

A significant challenge – likely a candidate for an entire rewrite. Code is usually poorly maintained and low quality, with possible security issues and significant variation in style. Documentation poor or non-existent.

AMBER · 10–19

Issues to address before new work, but probably no full rewrite. Expect some “workarounds” and possible security issues – written well initially but poorly maintained. Documentation limited but useful.

GREEN · 20–30

Generally well developed and maintained. No issues, or only minor ones. Code well patched, secure and performant with no style variation. Documentation to a good standard and up to date.

04 — WHY LISTEN

WHY SHOULD YOU LISTEN TO US?

We've been in business for over 18 years, and there's a reason for that — we're consistently the very best at what we do. We've worked with hundreds of clients on hundreds of projects. But don't just take our word for it:

"We've been using KOMODO on and off for 10 years and I can't think of a better client reference than that."

ROSS COONEY — CTO, IAMPROPERTY

"KOMODO are very refreshing: a talented, engaging and flexible team completely focused on delivering the customer's objectives."

PAUL CRENNELL — CTO, ORCHARD SYSTEMS

ASSESSMENT CRITERIA — 01 / 10

TECHNOLOGY AUDIT ASSESSMENT CRITERIA

01 · Project Structure

We all want to deliver on time. When a project is cleanly structured and well organised, it becomes obvious why it plays such an important role. Managed correctly, structure improves communication within teams – which in turn improves efficiency and speed of delivery.

---

0 / 3

The product follows no observable structure. There is no dependency management, and files are placed randomly.

1 / 3

A structure exists, but files are duplicated or often in the wrong location. Limited or no dependency management. It's difficult to infer where an executing piece of code sits.

2 / 3

A structure is implemented and mostly followed. Dependency management exists but may not be uniform or idempotent. You can infer where executing code is located.

3 / 3

An industry-standard structure is closely adhered to. Dependencies are locked; files separated by type/concern/layer (models, helpers, services). No ambiguity in a module's role; test files marked; tooling and config visible; documentation aids clarity; structure templated through automation and built platform-agnostic (e.g. cross-env on NodeJS).

ASSESSMENT CRITERIA — 02 / 10

02 · Tooling

Tooling is an essential part of modern software engineering – crucial for quality assurance, automating tasks and beyond. Sufficient tooling reduces project risk through automated checks and fixes, and helps developers quickly gain knowledge of a system.

The question is: does suitable tooling exist for the project – present, configured, runnable, and ideally tied into workflow stages like Git commit hooks?

---

0 / 3

No project tooling or automation at all. Tasks are done manually by hand and regularly repeated. No development environment is provided.

1 / 3

Limited tooling. Some repeated tasks have scripts. No automated code-quality control. A dev environment is provided but needs significant configuration. No CI/CD process.

2 / 3

Some tooling – in particular CI/CD is employed. All repeated tasks are scripted. Some linting and automated quality control. A dev environment is provided and functional.

3 / 3

A dev environment fully encapsulated within the project, plus a suite of tooling: database setup/teardown/seed, linting, package manager, build processes, test runners, CI/CD, dependency and security checkers, quality analysis and formatters. Documentation explains what exists and why.

ASSESSMENT CRITERIA — 03 / 10

03 · Design Patterns & SOLID Principles

Design patterns are abstract solutions to common problems. Each implementation can vary, but using them lets us solve problems in consistent, reliable, maintainable and expandable ways. When SOLID principles are applied, systems are easier to maintain, expand and modify.

---

0 / 3

No observable design patterns employed. SOLID principles are not in place; the code is WET.

1 / 3

Inconsistent application of standards with limited engineering best practice. Adheres to SOLID in places, fat files in others. Limited dependency injection, inversion of control or separation of concerns. Framework-provided patterns used only occasionally.

2 / 3

A defined structure adhering to SOLID and using design patterns. Obvious signs of dependency injection, and the project is generally uniform.

3 / 3

A clear, consistent, well-documented architecture, identifiable in the code. Fully conforms to SOLID with a sensible selection of documented design patterns. Dependency injection used effectively with sensible inversion of control, and universal uniformity – every part sticks to its designated pattern, with no edge cases tacked on.

ASSESSMENT CRITERIA — 03 / 10

SOLID Principles

Single responsibility · Open–closed · Liskov substitution · Interface segregation · Dependency inversion.

Photo by Clem Onojeghuo

CLIENT STORY

“I asked KOMODO to help with an early-stage startup project to get some strategic thinking behind the plans. Not only were they a really nice bunch to work alongside, they flagged an early issue that would have cost us dear if not found. I had no experience of the development world, and they made it simple, easy and as jargon-free as possible. I’d highly recommend KOMODO for development and design expertise – plus good commercial logic, which is vital too.”

HILARY MINES — FOUNDER, TRUNDL UK

ASSESSMENT CRITERIA — 04 / 10

04 · Tests

Automated testing lets us repeatedly exercise code so that as we change it, it keeps functioning as intended without regressions. Manual testing against a robust plan lets humans assert the software works, without behavioural, display or usability quirks. Testing should happen from the get-go – de-risking upgrades and speeding up development.

Did you know: in 1998 NASA lost the Mars Climate Orbiter to a failure to convert imperial to metric units – a $125m loss that software testing could have prevented.

---

0 / 3

No testing, in any form, features in project processes.

1 / 3

Limited automated testing with some scattered unit tests. Limited static or dynamic analysis. No documented manual test plan.

2 / 3

A suite of unit tests covering the main happy-path elements. Some static and dynamic analysis in use (formatters, commit hooks, npm scripts, or documented). A manual happy-path test plan exists for QA, though it may be out of date.

3 / 3

A full suite of static and dynamic analysis tools actively in use. Dynamic testing covers happy and sad paths, achieving >80% coverage. Static analysis baked in via commit hooks and formatters. Exclusions documented with reasons. A robust, up-to-date manual test plan can be executed by QA.

ASSESSMENT CRITERIA — 05 / 10

05 · Developer Documentation

A well-documented system allows for efficient communication, management and enhancement. Documentation is an absolute necessity for any software project – it helps fight off unwanted hitches like the accumulation of technical debt.

See how we averted a documentation disaster at Elanders – view the case study.

---

0 / 3

No project documentation whatsoever.

1 / 3

A readme with instructions to install, test, build, run and deploy the project. No other documentation exists.

2 / 3

Some extended documentation detailing the project's design – most likely created during early design phases, but perhaps not maintained or updated since.

3 / 3

Several documents clearly showing the design and implementation. Detailed rationale for the chosen design, especially around unique choices. Consideration given to future progress and potential changes.

ASSESSMENT CRITERIA — 06 / 10

06 · Revisioning

Code revisioning helps maintain workflows and product stability. Git is most commonly used these days, but it doesn't have to be.

---

0 / 3

No source control or revisioning is in place.

1 / 3

File or repository versioning is apparent; changes made are typically documented within the repository.

2 / 3

Source control is employed, but commits may be bulky and difficult to parse. No merge strategy. Insufficient or no pull-request workflow.

3 / 3

All code is revisioned from the beginning, commits kept as small as possible and clearly commented. Each feature or bug fix is worked on in its own branch, pull-requested and fully peer-reviewed before merging. Git repositories adhering to GitFlow are current best practice.

ASSESSMENT CRITERIA — 07 / 10

07 · Case Style, Naming Convention & Code Style

When we write code in a consistent style – using a consistent naming convention and case style – we reduce the complexity overhead by depending on expected meanings and structures.

---

0 / 3

No standards have been employed at all.

1 / 3

Case style and naming convention have uniformity, but code style varies.

2 / 3

Code style, naming convention and case style are applied consistently throughout. The style may not match industry defaults, but its application is universal and understandable to developers.

3 / 3

Case style and naming convention are applied consistently throughout. Conventions are obvious to the reader – e.g. prefixing an interface for business or domain logic where terminology is applied in variable and function names. Best practice is to provide, or link to, a glossary.

ASSESSMENT CRITERIA — 08 / 10

08 · Security

Security is of paramount importance to a software project. Following security best practices helps minimise the risk of long-term issues.

Did you know: cybersecurity threats cost the economy around 0.8% of global GDP (~$600bn). Per McAfee and CSIS, 32% of UK businesses suffered a data breach in 2019 – an average annual cost of £4,180, before long-term impacts to productivity or reputation.

---

0 / 3

There are critical security vulnerabilities in the application.

1 / 3

Known vulnerabilities that will require significant investment to address. Application security has been considered while building the product.

2 / 3

At worst, minor security concerns that are easy to tackle. Generally the application code security is in a good state.

3 / 3

No security concerns; best practice throughout. Secrets in environment variables (never committed). Security analysers and vulnerability checking built into workflows. Passwords never in plain text. No raw SQL – a query builder or ORM guards against injection. All inputs validated; access by permission not exclusion; CORS correctly configured; front-end protected against XSS.

ASSESSMENT CRITERIA — 09 / 10

09 · Appropriate Usage & Code Quality Concerns

Using frameworks, design patterns and technologies correctly has a huge impact on a project – best practice greatly improves efficiency, performance and maintenance. The number of visible issues within a site is a good indicator of how well maintained and supportable it is going forward.

---

0 / 3

The application is full of code quality concerns and exhibits no appropriate usage of frameworks, libraries or code.

1 / 3

The application does have code quality concerns that need addressing – but there are no red flags over how anything is used. (E.g. you wouldn't use a full validation library just to check a 4-digit number is between 1000 and 3000.)

2 / 3

Application code quality and appropriate usage are reasonable, as required by its function.

3 / 3

All frameworks, design patterns, packages and technologies are used in appropriate, clear ways. Unique usage is justified in documentation explaining why it was selected as a solution.

ASSESSMENT CRITERIA — 10 / 10

10 · Roadmap & Maintainability Concerns

Are you aware of any issues with libraries being deprecated – platforms or frameworks beyond end-of-life, or system-level dependencies that may be removed in future? Such items present a significant challenge to long-term support and can endanger a project's overall chance of survival.

---

0 / 3

There are fundamental and irrevocable issues preventing any further iteration on the project.

1 / 3

The application and platform are fundamentally built on a legacy foundation. There is a real concern for the future viability of the project.

2 / 3

The application is in a reasonable state; whilst there are issues, it should be able to iterate on the project and address them.

3 / 3

The application is in a strong state, future-proofed even. There are no causes for concern and it can be iterated on immediately with great effect.

06 — TL;DR

TL;DR

Just skimming? Yeah, this is a big document. We know you've seen lots of downloadables like this – but every bit of our technical audit is packed full of helpful, actionable insights. Grab a coffee and enjoy.

A partnership to help your team deliver exceptional results

At KOMODO Digital we focus on creating exceptional results – going above and beyond to leave the technology in a better state than we found it. We take demanding roadmaps, complex user needs and fast-moving markets and transform them into digital products that work for you and your users.

We're used to taking on existing projects and making them work. Whether you need extra resource or have a mounting backlog, we can help – integrating with your in-house team as an extension of your capabilities to speed up your product workflows.

We design and develop with the user in mind. It's not just about meeting the brief – it's about exceeding expectations and delighting users while driving commercial growth. We combine and configure our offering to meet your specific challenges.

AUDIT CHECKLIST

AUDIT CHECKLIST

Score each criterion from 0 to 3, then total for your RAG rating.

ASSESSMENT CRITERIA	0	1	2	3
1. Project Structure	☐	☐	☐	☐
2. Tooling	☐	☐	☐	☐
3. Design Patterns & SOLID Principles	☐	☐	☐	☐
4. Tests	☐	☐	☐	☐
5. Developer Documentation	☐	☐	☐	☐
6. Revisioning	☐	☐	☐	☐
7. Case Style, Naming & Code Style	☐	☐	☐	☐
8. Security	☐	☐	☐	☐
9. Appropriate Usage & Code Quality	☐	☐	☐	☐
10. Roadmap & Maintainability	☐	☐	☐	☐

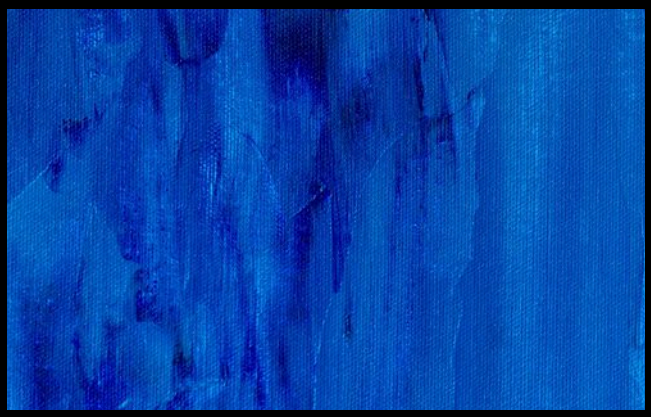
RED · 0–9 AMBER · 10–19 GREEN · 20–30 TOTAL / 30

OUR SERVICES

OUR SERVICES LIST

Looking for help creating a digital product that captures customer attention? Work with KOMODO and discover our expert-led services – from app and web development to captivating design.

USER RESEARCH



DIGITAL DESIGN – UX & UI

WEB DEVELOPMENT

APP DEVELOPMENT

KOMODO

GET IN TOUCH

BOOK A RAPID, COST-EFFECTIVE AUDIT.

We understand you're busy – so book our team to perform an audit for you. Conducted rapidly and cost-effectively, it'll help you discover opportunities to improve your current software and get ahead with your product roadmap.

GET IN TOUCH

WWW.KOMODODIGITAL.CO.UK

hello@komododigital.co.uk
0191 228 6555