We build the software **your team will inherit** — and write it so they can.

Architecture, backends, frontends, websites, maps, design systems and more than one language — from Copenhagen.

Engineer company
Software

Software built to be maintained, not just delivered

+45 50 10 61 84 · Copenhagen, Denmark

Carefully Crafted With Love by D.

Backends, frontends and the architecture that holds them together, written so that the next person to open the repository can change it. The work records describe the starting situation, what was done and the result.

Start with one workflow, integration or part of an application. We can agree a small delivery with acceptance checks, documented interfaces and a handover that enables your team to maintain the software.

Software work · Pricing and engagement options · Discuss your project

Start with the result you need

Tell us what your business needs to achieve. We connect software, data and infrastructure around the people who use them.

Explore our services Get in touch →

See how we approach the work

Find a service that matches your goal and see examples of what was delivered and how.

Cases Get in touch →

Read the work **in its context** — the decisions, delivery and reported outcomes.

See the whole catalogue

One partner for **11 services** from the architecture to the last screen.

Browse the catalogue

{ Software Engineering }

Deciding how the whole thing should be built

Platform & Solution Architecture

The shape of a system worked out on paper first, so the parts fit and the cost is known before anyone writes code.

Adopted Swift 6 strict concurrency over C

Adopted Swift 6 complete strict concurrency with no actors, bridging a blocking C event loop to the main actor through one producer, one consumer and one ordering — after establishing that a task per event loses the ordering the interface depends on.

A bounded event pipeline for a macOS app

Bounded event buffers and logging after a recorded macOS incident with 41 MB per second growth and 111 GB of compressed pages on a 36 GB machine; a recorded full test run reduced core log lines from 610,996 to 1,411.

Built a native macOS client in Swift 6

Built a native macOS messaging client in Swift 6 and SwiftUI — 11,141 lines across 53 files — over the C interface of a Rust core linked as a static archive from a pinned revision.

See all 20 achievements

How we work

Understand first

We learn the problem before we propose a solution. The brief you arrive with is the start of that conversation, not the end of it.

Build it to last

Durable over quick, clarity over cleverness. The system still has to be workable by the people who inherit it.

Stand behind it

We agree any ongoing operation or support as part of the engagement. Handover makes responsibilities and the next contact route clear.

{ Software Engineering }

The engine behind your app or website

Backend & API Development

The part users never see: where the data lives, what the rules are, and how other systems are allowed to talk to it.

Reached the unused half of the core

Reached the half of the messaging core the application had never used — backup transfer, disappearing messages, message editing and resending, verified invitations, proxies and encryption policy — driving every test against the real library with no mocks.

Wrote a parser verifying every FFI call

Wrote a parser that reads the real 7,308-line C header and verifies every call site, every enum constant and that every pointer-owning class is final, after a hand-written placeholder header let calls to three removed functions compile, link and crash.

Adopted Swift 6 strict concurrency over C

Adopted Swift 6 complete strict concurrency with no actors, bridging a blocking C event loop to the main actor through one producer, one consumer and one ordering — after establishing that a task per event loses the ordering the interface depends on.

See all 32 achievements

{ Software Engineering }

The screens people actually use

Frontend Development

Interfaces that load quickly, work on a phone and behave the same in every browser your customers happen to use.

Built a 48-token macOS design system

Built a 48-token design system on the platform's own glass material and wrote the linter that rejects a magic number, a hardcoded font size, or an animation that ignores the reduce-motion preference.

Named all 17 icons for screen readers

Named every icon-only control in the interface for screen readers after finding the send button announced as "arrow up circle, button", and wrote the linter that requires the label within eight lines of the icon.

A bounded event pipeline for a macOS app

Bounded event buffers and logging after a recorded macOS incident with 41 MB per second growth and 111 GB of compressed pages on a 36 GB machine; a recorded full test run reduced core log lines from 610,996 to 1,411.

See all 30 achievements

{ Software Engineering }

A working product, end to end

Full-Stack Product Development

One team taking an idea from a sketch to something your customers can log into, instead of handing it between three suppliers.

Built a native macOS client in Swift 6

Built a native macOS messaging client in Swift 6 and SwiftUI — 11,141 lines across 53 files — over the C interface of a Rust core linked as a static archive from a pinned revision.

Built a database-driven CV generator

Built a database-driven CV, references, portfolio and cover-letter generator in Python — 41 modules, 10,580 lines — rendering six output formats from one 23-table SQLite source assembled by a 19-step idempotent pipeline.

Built a multilingual Hugo site in 3 languages

Built a multilingual static site in Hugo across 108 template files including 53 partials, publishing every page in four representations from one content tree in three languages, and again under seven focused subdomains built from that same tree.

See all 16 achievements

{ Software Engineering }

A website you can update yourself

Website Development & CMS

A site that is quick, findable and editable by your own staff, without a bill for every change of wording.

Published a readable Tor onion mirror

Published the site as a Tor onion mirror on a readable address beginning engineer, with the key generated off the host so it never reached this repository or a laptop, and left it the one front door of four that is never counted.

Built the visual identity from 2 drawings

Built the company's icon and favicon sets from two hand-made drawings, with every derived asset regenerated by script and a check that fails when a derived file was committed before its source.

Passed 106 of 106 PDF/UA-1 profile checks

Recorded PDF/UA-1 validation across nine documents with 106 of 106 configured accessibility-profile checks passing, while the archival variant failed one of 146 rules.

See all 22 achievements

{ Software Engineering }

Showing up when customers ask an AI

AI Agent Readiness

A site or an app that the AI assistants your customers use can find, read and act on, checked on every release rather than promised once.

Ran development as 51 written initiatives

Ran the site's development as 51 written initiatives across 54 plan documents, each carrying its open boxes, its measurements and the decisions it declined — including six of eight agent-readiness findings refused with the reason recorded.

Fixed a sitemap with one shared timestamp

Fixed a sitemap where 172 of 176 URLs shared one modification timestamp, by taking the date from git history after establishing that the export rewrites every file on every run.

Replaced 963 metadata blocks with one graph

Replaced 963 anonymous structured-data blocks that restated the company 1,671 times with one linked graph of 16 types minted from stable origin identifiers.

See all 5 achievements

{ Software Engineering }

Letting AI use the software you already have

AI Agent Tools & MCP Servers

Your existing software opened up as tools an AI agent can call, without rewriting anything that already works.

Built the operator back-office for the platform

Built the operator back-office — around 40 admin routes and 128 components covering claims, moderation, feature flags, cache and diagnostics — so the platform can be run without database access.

Moved slow work onto a River job queue

Moved slow work off the request path onto a River job queue — 15 worker modules, 8 scheduled tasks and 20 pg_cron jobs — so a request returns while the work behind it carries on.

Generated the API contract from the database out

Generated the API contract outward from the database — OpenAPI, a 44,076-line typed TypeScript client, 61 mock handlers and the limits the UI enforces — with a guard at every hop that fails on drift.

See all 5 achievements

{ Software Engineering }

Maps that answer questions

GIS & Geospatial Solutions

Anything that happens in a place — vessels, vehicles, plots, sensors — put on a map you can search, measure and share.

A mapping product, carried end to end

Architected, developed and supported infrastructure, data processing and a map application over 2 years, covering both software delivery and operational maintenance.

Led 30+ GIS projects from build to support

Led the development, deployment, and support of over 30 GIS projects, demonstrating expertise in PostgreSQL, Bash, Python, JavaScript, GDAL, ArcGIS, PostGIS, and Mapbox technologies.

Managed 30 successful GIS projects to delivery

Managed the execution of 30 successful GIS projects, demonstrating leadership in delivering innovative solutions across the field.

See all 13 achievements

{ Software Engineering }

Making it obvious how to use

UI/UX Design & Design Systems

Layout, wording and flow arranged so people finish what they came to do instead of ringing your support line.

Built the visual identity from 2 drawings

Built the company's icon and favicon sets from two hand-made drawings, with every derived asset regenerated by script and a check that fails when a derived file was committed before its source.

Rewrote error messages to stop blaming users

Rewrote the application's error messages against a written tone standard after a refused sign-in blamed the user for mistyping when the provider actually required an app-specific password, and covered it with a test that names the provider.

Built a 48-token macOS design system

Built a 48-token design system on the platform's own glass material and wrote the linter that rejects a magic number, a hardcoded font size, or an animation that ignores the reduce-motion preference.

See all 20 achievements

{ Software Engineering }

Working in more than one language

Internationalization & Localization

Software that reads naturally in every market you sell in, including the dates, the currencies and the way names sort.

Built a cross-language figure check

Built a cross-language content check that fails when a translation drops a figure the English states, and when a language uses notation it does not use — finding two Danish descriptions missing a metric and sixteen Ukrainian spans quoting in the English style.

Moved every string into a content tree

Moved every user-facing string out of Python into a content tree of 874 files across three languages, after finding dead translations nobody could see were dead and a check silently grading a third of the achievements.

Built a database-driven CV generator

Built a database-driven CV, references, portfolio and cover-letter generator in Python — 41 modules, 10,580 lines — rendering six output formats from one 23-table SQLite source assembled by a 19-step idempotent pipeline.

See all 8 achievements

{ Software Engineering }

An AI assistant that knows your trade

Custom AI Agent Development

An agent wired to the systems you use, tested on cases your own experts write, overseen by your people and never tied to one model.

Made six collectors state what they skipped

Made six reporting collectors state the part of their input they never read — files that stayed shut, probes not installed, journal lines unrecognised — and fitted thresholds to 116 measured samples, after establishing that a blind collection and a healthy host render the same page.

Wrote tests for the checkers themselves

Wrote tests for the checkers themselves after establishing that a checker fed only clean input will one day report clean because it read nothing — planting a misspelling to confirm the spell-check finds it, and taking an id range from the database rather than from a number in the test.

Built a 22-linter commit gate

Built a commit gate of 22 one-line linters plus five that earn a paragraph, with no warning tier and no inline suppressions permitted, covering HTML, CSS, JavaScript, Python, YAML, Markdown, shell, links, spelling, secrets and typography.

See all 5 achievements

We work with **the platforms** your business already runs on.

Apple Microsoft DigitalOcean AWS Google

See the whole stack

Ready for the AI agents your customers send

Found

A robots file that says what agents may do, a sitemap with true dates, and every change announced through IndexNow the day it ships.

Read

A clean Markdown copy of every page, an llms.txt map for language models, and structured data that says what you do.

Usable

Accessible forms and controls, plus a described API for the application operations selected for agent access.

See how it works Ask about your site →

AI agents built for the work your trade does

Connected

Wired to your ERP, CRM, booking system and mailbox through described tools, each with one narrow job.

Measured

Scored against real cases your experts write, so every change to a prompt, a tool or a model is tested before it ships.

Supervised

What it does alone and what a person approves, written down with you, enforced in the tools and traced step by step.

See how agents are built Ask about your trade →

The software you run, made into tools agents can call

Unpicked

The rules hidden in screens, forms and stored procedures found, written down and moved into one layer.

Exposed

Each operation offered as a typed, described tool over an API or an MCP server, acting only within its user's rights.

Kept running

A scoped integration can expose one operation at a time alongside existing screens; application changes are agreed after reviewing its interfaces.

See how software opens up Ask about your software →

Tell us what you are building.

No form, no funnel, no sales sequence — just one e-mail to a person who will read it.

Start the conversation

This address shows the software work only. The company also works with data, runs infrastructure and advises teams — that side of it is at engineer.company.
See everything Engineer ApS does →

Begin with a focused improvement

A workflow, a data flow or a system that needs improvement can be a useful first task. We can agree its scope together.

See what it costs Get in touch →

Plan the first conversation

Tell us the result you need, what is already in place and when you need it. We can use that to discuss the next step.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

On the server, we work with **open-source Linux** — no licences to pay for, and nothing hidden.

Ubuntu Debian

How we run servers

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://software.engineer.company/