Engineer Software

Frontend Engineering

42 achievements

Connect the parts of your system

The work collected under “Frontend Engineering” shows one part of a working system. We can help connect that part to the rest of your operation. Interfaces that load quickly, work on a phone and behave the same in every browser your customers happen to use.

Explore our services Get in touch →

Find a comparable challenge

Follow an example that resembles your situation. Its review describes the decisions behind the result, beyond the technology label. Directed full-stack GIS map development — full-stack mapping, from database to browser

Directed full-stack GIS map development Get in touch →

Directed full-stack GIS map development

Directed full-stack GIS map development, overseeing PostgreSQL, Mapbox, ReactJS, and NodeJS to deliver an integrated solution.

Presented 200 UI/UX improvements

Presented 200 UI/UX improvements for the GIS map application, boosting revenue by 10x through enhanced software features.

Contributed to user interface design processes

Contributed to user interface design processes, ensuring intuitive, visually appealing, and user-friendly project interfaces.

Developed 100 web applications using HTML/HTML5

Developed 100 web applications using HTML/HTML5, CSS/SCSS, Django, WordPress, and Joomla, ensuring diverse online presence.

Designed 30 websites from Photoshop to HTML

Designed 30 websites, delivering unique and flexible solutions by converting Photoshop designs to HTML.

A maritime platform with clear system boundaries

Architected a layered maritime platform separating a Next.js PWA frontend, a Go (Huma/Fiber) API, and a PostgreSQL function layer, keeping all business logic in the database.

Built organization context switching for users

Designed an organization context-switching system with client localStorage and server-side cookie mirroring, letting users act as managed organizations while enforcing least-privilege authorization.

Built the Next.js 16 frontend: SSR, SSG and CSR

Built the Next.js 16 frontend with deliberate SSR, SSG and CSR strategies, and a reusable prefetched-server-page factory that removes the N+1 fetch cascade from each authenticated page moved onto it.

Delivered full Progressive Web App support

Delivered full Progressive Web App support — installable and offline-capable — with Workbox runtime caching via next-pwa.

Designed an iOS 26 'Liquid Glass' design system

Designed an iOS 26 'Liquid Glass' design system and a canonical component inventory enforced by lint rules to prevent UI divergence.

Designed the product's UI and UX end to end

Designed the product's UI and UX end to end — directory grids, dual card/table views, live requirement validators and breadcrumb navigation.

Hardened the application with nonce-based CSP

Hardened the application with nonce-based CSP, HSTS, SameSite cookies, least-privilege database roles and server-side entitlement re-checks.

Built an English/Danish i18n system

Established an English/Danish internationalization system carrying 12,027 messages per locale across 458 namespace files, with linter-enforced vocabulary and a 400-line budget per file.

Set the platform's founding technical decisions

Set the platform's founding decisions in the weeks after the repository opened in November 2025 — the layering, database-first data access and the zero-warnings bar — and they still hold nine months on.

Built and maintained Django apps for IPTV

Developed and maintained Django web applications for the IPTV platform, shipping new features and improving performance and stability.

Maintained a Hugo site and improved product UI

Maintained and enhanced the legacy Hugo static-site website while contributing UI/UX improvements to the primary asset-management product.

Combined web development with SEO and content

Drove client web success by combining custom website development and design with SEO, content strategy and copywriting.

Built a test suite across Go, SQL and browser

Built a layered automated test suite — 981 Go tests, 543 frontend and browser specs, 494 SQL behavioural tests — with mutation testing, property-based tests and an accessibility gate.

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.

Built company ownership claims and adjudication

Built company ownership claims end to end — a user claims a company, an administrator adjudicates, and an approval rewrites the authorization graph that decides who is allowed to edit what.

Built the social layer: posts, feed and groups

Built the platform's social layer — posts, feed, groups, mentions, a follower graph and an occasions digest — on the same function-first data layer as the rest of the product.

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.

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.

A safety net that no longer costs half an hour

Cut the site's browser-driven quality gate from 1,636 seconds to 615 by scheduling its checks longest-first through a worker pool bounded to four lanes, after measuring that alphabetical order cost 320 seconds against 224.

Replaced 27 font sizes with a 6-step scale

Replaced 27 rendered font sizes whose nearest neighbours were 0.6% apart with a six-step Major Third scale, and wrote the linter that fails the twenty-eighth.

Accessibility checks across 32 pages and both themes

Implemented WCAG 2.2 Level AA checks across 32 representative pages in both colour themes, with a conformance record, one AAA criterion checked and the partial result for criterion 1.4.13 stated explicitly.

The accessibility faults your checker misses

Found and fixed 11 accessibility defects the browser reported as healthy — eight footer links still in tab order behind pointer-events, a scroll timeline clipping the colophon off three pages, and a rule engine running 70 of its 105 rules.

Generated 495 pages from a SQLite export

Generated 495 achievement pages across three languages from a read-only SQLite export, with the page address authored as data so that correcting a sentence no longer moved the page and broke the link.

Closed the colour system at 28 colours

Closed the colour system at 28 documented colours with a linter that fails a value painted but undocumented, documented but unpainted, mis-measured, respelled as a literal, or within a perceptual distance of 0.02 of one already there.

Found the light theme missing a whole layer

Found that the light theme had been missing a full-page overlay layer since the day it was written, by asserting that both themes paint every layered surface with the same number of layers.

Made the reduced-motion setting honest

Made reduced-motion honest after finding eleven selectors still animating under the preference, because a universal transition-none rule loses on specificity to any rule carrying a class.

Built a 637-line print stylesheet

Built a 637-line print stylesheet against four documented rendering-engine behaviours, including run-in headings that printed white on white for any reader whose browser preferred dark.

Mirrored the site to Gemini and Gopher

Built Gemini and Gopher outputs from the same website source; the achievement records 777 documents in each format at that stage, with zero HTML bytes changed by adding the formats.

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.

Cut the stylesheet bundle from 87 KB to 31 KB

Cut the stylesheet bundle from 87 KB to 31 KB by taking a base64 font out of it, and dropped 756 KB across 22 files that were published on every deploy and referenced by nothing.

Linted the 10,242-line quality gate itself

Brought 10,242 lines of quality-gate JavaScript under a formatter and a linter after establishing it was the largest body of code in the repository and the only one nothing read, fixing 13 findings and suppressing none.

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.

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.

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.

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.

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.

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.

Focus the engineering effort

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Put the subject in your context

Tell us how “Frontend Engineering” appears in your own system. We can help choose the relevant service and define the work around it.

Explore our services

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

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/categories/frontend/