Building My Own Car for the Race

Seventeen Years in Project Management, and Why I Wrote My Own Software Rather Than Buy It [1]

Naval Singh

Founder, NavalPM

Abstract

The Project Management Institute's July 2026 refresh of the PMP certification distinguishes between perishable skills tied to a specific tool, durable skills and frameworks that transfer across tools, and enduring capabilities — judgment — that compound across a career. This commentary tests that distinction against a specific career that runs an unusual course: infrastructure execution on the ground, a foundational project management certification earned while working full-time, a decade inside large consulting and enterprise-platform organizations, a mid-career return to formal technical education in data science and machine learning, and finally, in 2026, the decision to build an independent project management software company rather than adopt an existing one. That last decision is the spine of the piece: choosing to build the tool rather than buy it is, in miniature, a wager that the underlying discipline of project management is better served by software designed around durable judgment than by software optimized for the metrics of a much larger company. The commentary traces each stage of that career for what it actually contributed — separating what turned out to be perishable from what turned out to be durable — and uses the answer to explain specific architectural and design decisions made while building the product, including its AI governance model, its native support for scaled-agile planning, and its approach to data portability. It closes with an argument that direct practitioner experience, including years spent on the vendor side of the very platforms practitioners depend on, is an underused source of product design authority in an industry increasingly built by people who have never run the projects the software is meant to serve.

[1] How to cite this article: Singh, N. (2026). Building My Own Car for the Race: Seventeen Years in Project Management, and Why I Wrote My Own Software Rather Than Buy It, story, *PM World Journal*, Vol. XV, Issue IX, September.

Keywords: *project management software; PMP certification; durable skills; career transitions; enterprise transformation; ServiceNow; data science education; AI governance; founder journey; accessibility*

Introduction: Why Build the Car

There is an old argument in motorsport about why a team builds its own car rather than buying a chassis and an engine already proven to win. The bought car is faster to the grid and cheaper to field. The built car costs more, takes longer, and might not even be faster in its first season. But the team that builds understands every part of what they are racing, can fix what breaks without waiting on a supplier, and — this is the part usually left out of the romantic version of the story — actually owns what they learn in the process. I use this as a metaphor deliberately, not as a description of an actual project: I have never built a race car. But I have spent much of the last two years building a project management platform from scratch, one architectural decision at a time, when adopting Jira or Asana or monday.com and moving on with my life was sitting right there as the obviously cheaper option. This piece is an attempt to explain why I built the car instead, using the seventeen years before it — on infrastructure sites, in consulting rooms, inside an enterprise platform vendor, and, more recently, back in a classroom learning to actually write the software myself — as the evidence for why that decision was not impulsive.

The Project Management Institute's own framing, introduced with its July 2026 PMP refresh, gives me useful language for this. PMI's VP of Learning, Dr. Kelly Heuer, describes professional capability as a spectrum running from perishable skills — the specific tools and platforms essential only for today — through durable skills and frameworks that travel intact from one tool to the next, to enduring capabilities: judgment and mindset that compound across an entire career. What follows is my attempt to sort my own career onto that spectrum honestly, and to show my work.

2005–2009: National Institute of Technology, Rourkela

My formal education began in electrical engineering, not computer science or business, and not project management. That ordering matters more than it might seem to. An engineering curriculum teaches you to reason about systems under constraint — load, tolerance, failure modes — before it teaches you anything about managing the people or budgets required to build those systems. I did not choose electrical engineering because I planned to manage projects; I chose it because it was rigorous and because Rourkela, an industrial town built around a large public-sector steel plant, made engineering feel like the practical, obvious path. It turned out to be excellent, if accidental, preparation for

what came next: infrastructure work does not care whether you have studied management yet. It cares whether you understand the system you are standing inside.

2009–2013: Project Engineer, Indian Oil Corporation

Infrastructure execution does not forgive abstraction. As a Project Engineer at Indian Oil Corporation, I managed procurement, vendors, and on-site crews of fifty or more people on live infrastructure — the kind of work where a scheduling mistake shows up as a stalled crew standing in a field, not as a red cell in a spreadsheet. There is a particular discipline that on-site execution teaches that a classroom cannot fully replicate: the difference between a plan that looks complete on paper and a plan that has actually accounted for the sequence in which real materials, real permits, and real people become available. I learned to distrust any schedule that had not been walked, literally, on the ground it described.

One recurring pattern from that period stayed with me longer than any specific project: the gap between what a status report said and what a site supervisor would tell you if you asked him directly. Reports tend to describe progress in terms convenient for the person writing them. A supervisor standing in the yard describes it in terms of what's actually blocking him. Nothing about that gap is unique to industrial infrastructure — it is the same gap, dressed differently, that shows up in a software team's sprint board versus what an engineer will tell you in a hallway conversation. I did not have language for that pattern yet in 2009. I would spend much of the rest of my career, and eventually a fair amount of product design work, trying to close it.

The Certificate That Didn't Expire: IGNOU, 2010–2011

In December 2010, a year into that job, I sat the term-end examinations for a Post Graduate Certificate in Project Management through Indira Gandhi National Open University — studying part-time, by distance, around a full-time site role. The certificate arrived a few months later: four courses — project fundamentals and appraisal, planning and organization, implementation and control, closeout — completed First Division with Distinction, 75 percent overall.

What strikes me most about that certificate now, fifteen years and one AI-driven overhaul of the entire PMP curriculum later, is how little of its actual content has aged. Appraisal, planning, implementation, closeout: none of those four words require a specific software product to be true. They are the same four words I would use today to describe what any competent project management tool should be structured around, NavalPM included. That is, in PMI's current language, about as clean an example of a durable skill as I can

offer: content learned in a distance-education program in 2010, on paper, with no software behind it at all, that turned out to still be the entire scaffolding of the discipline in 2026.

2013–2015: Indian Institute of Management, Indore

Going back to business school after four years in the field was, in part, an attempt to formalize what I had been learning by instinct on-site — to put language and frameworks around decisions I had already been making under pressure, without always being able to explain why they were right. An MBA does not teach you to run a project crew; I already knew how to do that. What it teaches, done well, is how to reason about a project's place inside a larger organization — capital allocation, stakeholder incentives, the politics of a budget line — the layer above the one I had been operating in at Indian Oil. It also gave me my first real exposure to case-based reasoning under ambiguity, which turned out to be more durable than almost anything else in the program: the specific frameworks taught in 2013 have mostly been superseded or renamed since, but the habit of reasoning through an unfamiliar situation from first principles has not.

2015–2016: Business Consultant, Infosys Consulting

My first role after business school was a short, dense stretch as a Business Consultant at Infosys Consulting — a deliberately transitional year, moving from the specificity of infrastructure execution into the more generalized world of enterprise consulting, working across client engagements rather than a single site. It was here that I first saw, from the consulting side rather than the execution side, how much of large-organization project failure has nothing to do with technical competence and everything to do with how information moves — or fails to move — between the people doing the work and the people deciding what work matters. That observation would recur, more sharply, at both of my next two stops.

2016–2020: Strategy/Business Consultant, IBM

At IBM, managing multi-account programs across cloud, analytics, logistics, and transportation, I owned scope, financials, risk, and cross-program dependencies across several projects that were each, individually, someone else's entire world. This is where I learned that most of the real failure modes in program management are not technical. They are governance failure modes: a stakeholder who was never properly aligned, a risk that was tracked but never escalated to the person who could act on it, a dependency that was known to one team but invisible to the team that actually needed to see it.

Nothing in that list requires a particular tool to fix. It requires a discipline of making the right information visible to the right person at the right time — and I watched, repeatedly, how much software either supports that discipline quietly or actively works against it through complexity, cost-gating features that should be table stakes, or simply burying the signal an executive needed inside a dashboard built for someone else's job. I did not yet know I would spend the next several years watching that same failure mode from the vendor's side of the table.

2020–2023: Senior Consultant, Transformation Programs, ServiceNow

Leading enterprise and government digital transformation delivery at ServiceNow put me on the vendor side of exactly the kind of large platform I had spent the prior years governing programs around — establishing governance cadence, executive reporting, and KPI frameworks, and coordinating internal teams, systems-integrator partners, and senior client stakeholders through platform implementations that, at their best, measurably improved service efficiency for the organizations adopting them, by roughly 20 percent on the engagements I led.

It also put me in a position to watch, repeatedly, what large enterprise platforms cost the organizations trying to adopt them — not just in licensing, but in implementation timelines, SI dependency, and the sheer organizational weight required before a mid-sized team could get meaningful value out of the platform. None of that is a criticism of any specific product; it is a structural feature of how enterprise software at that scale is typically sold and deployed, and I helped sell and deploy it, competently and in good faith, for three years. But it planted a specific, recurring question I did not yet have an answer to: what would it look like to build something with the same underlying discipline — real governance, real structured planning, real audit trails — without requiring an enterprise budget and an SI partner to get there? I left that question unanswered for a while. It did not leave me.

2021–2026: Learning to Build the Engine — BS in Data Science and Applications, IIT Madras

Somewhere in the middle of the ServiceNow years, I made a decision that looked, from the outside, like a detour: I enrolled in the Indian Institute of Technology Madras's BS in Data Science and Applications program, studying part-time alongside full-time consulting work, the same pattern I had followed a decade earlier with the IGNOU certificate. I am currently at the Degree level of that program, having completed the Foundation and Diploma stages.

This is the chapter of the story that actually explains the metaphor in this piece's title. A program manager who understands governance can specify what a project management tool should do. Being able to actually build it is a different, separate skill, and I did not have it yet. Two projects from that program turned out to matter more than I expected when I sat down, later, to design NavalPM. The first was a Kaggle-based competitive data science exercise on sentiment prediction from movie reviews, coursework for CS2008 — a useful, if modest, first exposure to the applied machine-learning pipeline: data cleaning, feature representation, model evaluation, the unglamorous plumbing underneath anything that looks like 'AI' from the outside.

The second was more directly formative: an extracurricular project I built called an AI Compliance Agent — a system that routes natural-language questions across multiple large language model providers, specifically Claude on Amazon Bedrock and LLaMA 3 via a locally hosted Ollama instance, chosen dynamically depending on the question. It uses LangChain to chunk, embed, and search policy documents with FAISS and sentence-transformer embeddings; a natural-language-to-SQL pipeline to query a 180,000-row structured database without hand-written queries; and a hybrid mode that combines both document search and structured data lookup for compliance questions that need both. I built the multi-provider routing in that project before I understood I would need almost the exact same architecture — provider health checks, automatic failover, a query type deciding which model actually answers — for NavalPM's own AI project assistant eighteen months later. The compliance framing of that earlier project also left a mark I did not fully appreciate until later: a compliance agent only works if the person using it can trust exactly what it is checking against and exactly what data it is touching. That is a governance problem before it is a technical one, and it is the direct ancestor of the consent-gating system NavalPM's assistant now enforces on every request, server-side, before any project data reaches a model.

I am naming this chapter explicitly because I think it is the part of the story most easily left out of a founder narrative, and the part that actually does the explanatory work the car metaphor is gesturing at. Choosing to build a project management tool instead of adopting one is not, on its own, an interesting decision — plenty of people who have never learned to build software make that same choice and end up either failing or hiring it out. What made it a real decision rather than a slogan was going back, mid-career, to formally learn the parts of the craft — data pipelines, embeddings, retrieval, model routing — that seventeen years of program governance had never required me to learn, and that I would have had no credible way to design around without having actually built smaller, worse versions of them first.

2026: Building the Car — NavalPM

NavalPM exists because I eventually stopped asking the ServiceNow-era question rhetorically and started answering it with code. It has been built without institutional backing — no outside funding, no team beyond myself for long stretches — which is a structural fact, not a badge of honor, and it shaped nearly every decision in the product. When there is no budget to out-market the incumbents, the only lever left is to be more honest about what the tool does and does not do, and to make the parts that matter most genuinely hard to find elsewhere.

Three decisions follow directly from the years described above, not from a generic sense that these are good ideas. Native support for Scaled Agile Framework Program Increment planning — treated as a first-class object rather than a paid enterprise add-on — comes directly from watching organizations at ServiceNow treat structured, cross-team planning as a premium concern rather than a baseline discipline; PMI's own research on why enterprise reinvention stalls names exactly this same gap, with CEOs citing a disconnect between planning and execution as their leading barrier to reinvention. Free, unlimited client-facing viewer seats come from years of watching stakeholder visibility get gated behind licensing tiers, the same governance failure mode I first learned to name at IBM. And the consent-gated AI assistant — every action visible and actively agreed to before any project data reaches a model, enforced server-side rather than buried in a blanket terms-of-service acceptance, with consent re-requested any time what is disclosed materially changes — comes directly from the compliance-agent project at IIT Madras, built roughly two years earlier, once I understood that trust in an AI system is a governance property, not a feature you bolt on afterward.

PMI's CEO, Pierre Le Manh, put the underlying claim plainly at the PMP relaunch: "Using AI tools on its own is not enough to deliver project success." I would extend that claim one step further, from the practitioner's judgment to the tool itself: a project management tool that does not actively protect a practitioner's judgment — through consent, through transparency, through genuine data portability rather than lock-in — is not a neutral instrument. It is quietly training the perishable-skill habits PMI's refreshed exam is now trying to certify against. Buying the fastest available car and racing it is a perfectly reasonable choice for most teams. I chose to build mine because I had, by 2024, finally accumulated both halves of what building it required: seventeen years of knowing what the car needed to do, and two of actually learning how engines work.

A Closer Look: What the AI Compliance Agent Actually Taught Me

It is worth slowing down on the compliance-agent project, because the lesson it taught me was more specific than 'AI routing is possible.' The system had to answer questions against two very different kinds of source material — unstructured policy documents and a large, structured database — and the naive approach, sending every question to the same model with the same context window stuffed full of both, was slow, expensive, and frequently wrong in ways that were hard to debug after the fact. The fix was not a bigger model. It was a routing decision made before the question ever reached a model at all: classify what kind of question this is, decide which retrieval path and which provider actually fits it, and only then generate an answer, with the retrieved evidence attached so a human could check it.

That is a governance decision disguised as an engineering one, and I did not fully recognize it as such until I was designing NavalPM's own assistant eighteen months later and found myself solving the identical problem: a project status question, a policy question, and a 'draft me a work breakdown structure' request are not the same kind of task, should not hit the same provider by default, and should not be answered with the same amount of visible reasoning. utils/modelRouter.js in NavalPM's codebase — which tries whichever configured provider fits a given question type, ordered by a live health/quota estimate rather than a fixed priority list, and falls through cleanly on failure — is, structurally, the second version of something I first built badly as a student project. The difference the second time was that I understood, from the compliance framing of the first version, that the routing logic itself needed to be auditable, not just functional: a PM asking NavalPM's assistant a question should be able to know, after the fact, roughly what happened to answer it, in the same way a compliance officer needs to know what a compliance agent actually checked.

What 'Durable-Skill-First' Design Actually Requires

It is easy to say a tool should support judgment rather than replace it. It is harder to say what that means in the specifics of a running product. A few commitments, beyond the three named above, follow from taking PMI's distinction seriously rather than treating it as marketing language.

Governance as infrastructure, not a feature flag

Every mutating action across every organization on the platform is captured in a structured, queryable audit log — who did what, from where, and when — not as a premium add-on but as a baseline property of the system, the same instinct that governed

how I ran executive reporting at IBM and ServiceNow. A tool that cannot answer 'what changed and who changed it' on demand is not really supporting governance; it is asking the practitioner to remember to do so manually, which is exactly the kind of erosion of durable discipline PMI's refreshed exam is trying to guard against.

Genuine portability, including the uncomfortable version

If durable skills are supposed to travel with the practitioner rather than staying trapped in one vendor's data model, the tooling around them should make that portability real — including a real migration path away from legacy tools, and including the version where a user migrates away from NavalPM itself as easily as they migrated in. That is a harder commitment to keep than it is to state, and I do not think most vendors, regardless of size, actually keep it.

Consent as a default, not a setting

An AI assistant that silently ingests project data to generate suggestions is optimizing for perceived magic at the cost of the practitioner's own situational awareness — precisely the erosion of judgment PMI's refreshed exam is trying to certify against. NavalPM's assistant requires explicit, visible consent before any project data leaves the system for a given kind of request, recorded server-side and versioned, so a change to what is disclosed re-triggers consent rather than silently expanding under an old agreement. It is slower and less impressive in a live demo than an assistant that just answers. It is also the only version of AI assistance I was willing to ship, having spent two years, by that point, thinking about compliance systems specifically as trust infrastructure rather than as a convenience feature.

Bring-your-own-model as an architectural boundary, not a marketing checkbox

A project can opt out of NavalPM's own AI infrastructure entirely and route every assistant query to a provider of its own choosing, with its own credentials, stored encrypted and never sent anywhere except to that provider. This matters more than it sounds: it means an organization with its own compliance requirements around where its data can go is not taking my word for it that NavalPM behaves responsibly — the architecture makes it structurally impossible for that project's data to reach NavalPM's own infrastructure at all once the option is enabled. That is a stronger commitment than a privacy policy, and it is the kind of commitment that is only obvious to build if you have spent time, as I had at ServiceNow, watching how much enterprise trust actually depends on verifiable architecture rather than stated intent.

Limitations and Open Questions

I want to close the practical half of this piece honestly rather than promotionally, because a commentary that only lists what a tool does well is not adding much to a discipline PMI is currently asking to prize judgment over polish. NavalPM, as of this writing, has real, known gaps. It has no enterprise single-sign-on beyond Google OAuth, and no SCIM-based user provisioning — both real requirements for larger regulated organizations, and both currently unbuilt. Its real-time collaboration is limited; two people editing the same board at the same moment do not see each other's changes seamlessly the way some larger, better-resourced competitors now support natively. It has no native mobile application. None of these gaps are secret, and none of them are dressed up as intentional restraint when they are, more honestly, a function of a one-person team's limited hours.

I am naming them here because I think the more interesting question this whole piece is implicitly asking — whether a career built on practitioner judgment translates into better product decisions than a career built on venture-funded growth targets — is only a fair question if it is asked against the tool's actual current state, gaps included, rather than an idealized version of it. Some of these gaps I intend to close; a few I may choose not to, if closing them would mean compromising the same judgment-first design principles this piece has been arguing for. That decision, too, is one only a founder with practitioner judgment behind them, rather than a growth target in front of them, gets to make honestly.

Why Accessibility Is Not a Side Argument

PMI projects that organizations will need up to 30 million new project professionals by 2035, at the same time as roughly half of today's technical skills risk becoming obsolete within five years. The market data PMI has published alongside the PMP refresh reinforces how much is riding on that gap closing: certifications granted in the first five months of 2026 rose 36 percent year over year, and PMI reports that PMP holders in the United States earn a median salary nearly 24 percent higher than uncertified peers. Those are strong incentives for an individual practitioner to get certified. They say much less about whether that practitioner will have access, afterward, to tools built to reinforce the judgment the certification is meant to represent, rather than tools priced and packaged for the organizations that already have the budget to onboard slowly.

I did not arrive at my own project management education through an employer-sponsored enterprise program — I earned it through an open university, while working full time on infrastructure sites, because it was the accessible path available to me at the time. A decade later, I re-learned the technical half of my current craft the same way, part-time,

through a public institution's distance program, while still working full time. Both facts shape how I think about pricing NavalPM now: a tool priced at a fraction of the enterprise incumbents' per-seat cost, with free trial access rather than a sales-call gate, is not charity. It is a bet that the discipline gets stronger when the people building the durable skills PMI now certifies for — solo consultants, small teams, career-changers, people early in the kind of path I started on in 2009 — can actually afford the tools built to reinforce that discipline, not just the credential.

It is worth being specific about what 'accessible' means in practice, because the word gets used loosely. For NavalPM it has meant three concrete pricing decisions rather than a mission statement: keeping a genuinely usable free/starter tier rather than a crippled trial designed purely to convert; refusing to gate client- and stakeholder-facing seats behind a paid tier, on the theory that visibility for the people a project actually serves should never be the thing an underfunded team has to cut first; and pricing the paid tiers against what a small team or solo consultant can plausibly justify, rather than against what an enterprise procurement process is willing to sign off on. None of these are radical ideas. They are simply harder to hold onto once a company has investors asking about average revenue per account, which is, not incidentally, one honest reason I have chosen to build this alone for as long as I can.

For Other Practitioners Considering the Same Bet

I do not think everyone reading this should go build their own project management software; that would be a strange and impractical lesson to draw from one person's career. But I do think the underlying pattern — spend enough years as a practitioner to know precisely where existing tools fail you, then go acquire, deliberately and often slowly, the specific technical skill required to fix exactly that failure rather than a generic one — generalizes further than founder biographies usually admit. Three observations from having done it, offered plainly rather than as inspirational advice.

The gap between knowing what's wrong and knowing how to fix it is a skills gap, not a vision gap

For most of the ServiceNow years, I could have described, in detail, what was wrong with how enterprise platforms gate value behind implementation cost. That description was not, on its own, useful to anyone. It became useful only once I had also spent two years learning to build retrieval pipelines and model routers well enough to design something different. A practitioner's frustration with existing tools is a real signal, but it is not yet a product, and mistaking the frustration for the capability to fix it is a common and costly error I tried hard not to make.

Formal, part-time, mid-career education is underrated as a founder pathway

Neither the IGNOU certificate in 2010–11 nor the IIT Madras program starting more than a decade later were full-time commitments; both were completed alongside full-time jobs, on the theory that the discipline of finishing something difficult while still delivering at work is itself part of what the credential is certifying. That path is slower and less visible than a career break for a full-time degree, and it is also, I think, more honest about how most working professionals actually have to acquire durable skills once they are a decade into a career with real financial obligations attached to it.

The vendor side teaches you things the client side cannot

Three years at ServiceNow taught me more about why enterprise software is priced, packaged, and sold the way it is than any amount of time as a client-side program manager could have. That knowledge is not flattering to the vendor side, and I do not think it is meant to be repeated as an accusation — it is simply what watching a business model from the inside teaches you, and it is a perspective founders who have only ever been customers of enterprise software tend not to have.

Conclusion: The Car Doesn't Have to Win Immediately

I do not think my career is unusual in the project management profession — most practitioners I have worked alongside, across a public-sector infrastructure project, two consulting firms, and an enterprise software vendor, have some version of this arc: hands-on execution first, formal grounding second, a slow accumulation of judgment that outlasts whichever tool happened to be standard issue at each stop. What is less common is going back, mid-career, to learn the specific technical craft required to act on that judgment directly, rather than writing it into a slide deck for someone else to build.

PMI has now drawn a formal line between what a tool teaches a practitioner and the judgment no tool can substitute for. I spent seventeen years on the practitioner side of that line, three of them watching it from inside a major enterprise vendor, and two more back in a classroom learning to build software myself, before trying to put something on the other side of it. A car built from scratch does not have to win its first race to have been worth building — it has to be honest about what it is, and it has to keep teaching the people who built it something true every time it is on the track. Whether NavalPM gets that balance right is still being tested in the market. But the argument, at least, was not written from a product roadmap. It was written from a work site in 2009, a certificate in 2011, a business-school classroom in 2014, a client engagement in 2016, an enterprise platform deployment in 2022, a data science course in 2024, and everything since.

Declaration of Interest

The author is the founder and sole developer of NavalPM, a project management software product discussed throughout this commentary as the practical outcome of the career described. This constitutes a direct financial and personal interest in the subject matter, and readers should evaluate the commentary's claims about NavalPM accordingly. Employers and institutions named (Indian Oil Corporation, Indian Institute of Management Indore, Infosys Consulting, IBM, ServiceNow, Indian Institute of Technology Madras) are described solely as part of the author's career and education history; no current relationship with, sponsorship from, or endorsement by any of them is implied or should be inferred. No other sponsorship, funding, or conflicting relationship influenced this submission.

Use of Artificial Intelligence

AI assistance (Claude, Anthropic) was used in drafting and structuring this manuscript from the author's own career history, direction, and factual input, including details drawn from the author's own IGNOU certificate, CV, and IIT Madras student profile as supplied by the author. The author reviewed, edited, and takes full responsibility for all claims, framing, and content in the final submitted version, consistent with PMWJ's Use of AI guidelines.

References

Project Management Institute. (2026, July 9). *In an AI economy, judgment is the new job security: PMI refreshes the PMP certification to validate the skills that outlast the tools* [Press release]. https://www.pmi.org

Project Management Institute. (2026). *PMI Pulse of the Profession 2026: Driving success in complex projects — from navigating tasks to navigating systems.* https://www.pmi.org

Project Management Institute. (2025). *PMI 2025 CEO Quantitative Survey: Manifesto for Enterprise Agility research.* https://www.pmi.org

Indira Gandhi National Open University, Student Evaluation Division. (2011). *Statement of Marks — Post Graduate Certificate in Project Management (awarded to the author, First Division with Distinction).*

Singh, N. (2023). *Sentiment Prediction on Movie Reviews* [IITM BS CS2008 course project]. Kaggle. https://kaggle.com/competitions/sentiment-prediction-on-movie-reviews

Singh, N. (n.d.). *AI Compliance Agent* [Software project]. GitHub. https://github.com/navalsingh9/ai-compliance-agent

About the Author

Naval Singh

Madras, India

Naval Singh began his career as a Project Engineer at Indian Oil Corporation (2009–2013), executing infrastructure and supply chain projects and managing on-site teams of 50 or more. In 2011 he earned a Post Graduate Certificate in Project Management from Indira Gandhi National Open University, completing the program First Division with Distinction. He completed an MBA/PGDM at the Indian Institute of Management, Indore (2013–2015), followed by roles as a Business Consultant at Infosys Consulting (2015–2016), a Strategy/Business Consultant at IBM (2016–2020) [title to confirm against internal records], and a Senior Consultant in Transformation Programs at ServiceNow (2020–2023) [title to confirm]. He holds a B.Tech in Electrical Engineering from the National Institute of Technology, Rourkela (2005–2009), and is currently completing a BS in Data Science and Applications at the Indian Institute of Technology, Madras, at the Degree level. In 2026 he founded NavalPM. Naval can be contacted at singhn9@gmail.com