THE AI-ERA CV

A new template for how you present your work

The old CV format — a list of companies, years, and job titles — doesn't tell anyone what you're actually capable of anymore. It never really did, but it especially doesn't now.

What actually matters today is this: what problems have you solved, how did you solve them, and what did that solving actually change for someone. Whether you used a team, an AI agent, or built it alone at 2 AM doesn't matter as much as the outcome.

This template flips the usual CV structure:

- 50% — Problems you've solved, written clearly: the problem, your solution, the resources you used, and the impact it had.
- 25% — Real, hands-on experience with cloud infrastructure (AWS, GCP, Azure — not just the one you're comfortable with).
- 25% — Your work history: companies, roles, years. Still relevant — just no longer the headline.

Two versions follow — one for senior folks, one for freshers/junior devs. Use whichever fits, or mix and match. There's no one correct format here — the point is to change how you're presenting yourself, not to follow this to the letter.

A practical note: this format works best for direct outreach, startups, and personal branding — situations where a human actually reads your CV. For applications through corporate ATS systems, keep a traditional chronological version alongside this one.

[Senior Developer / Solution Architect — your actual title]

email@example.com | +91-XXXXXXXXXX | linkedin.com/in/yourname | github.com/yourname | portfolio site

PROBLEMS SOLVED (THIS IS 50% OF YOUR CV)

Pick your 3-5 strongest examples. Real projects, real numbers, real links wherever you can.

PROBLEM 1

[one line — what was broken, slow, or missing]

Solution — *[what you built / the approach you took]*

Resources — *[team size, AI agents/tools, APIs, frameworks]*

Impact — *[time saved, revenue impact, users served, cost cut — real numbers]*

Live link — *[github.com/... or live URL, if available]*

PROBLEM 2

[one line — what was broken, slow, or missing]

Solution — *[what you built / the approach you took]*

Resources — *[team size, AI agents/tools, APIs, frameworks]*

Impact — *[time saved, revenue impact, users served, cost cut — real numbers]*

Live link — *[github.com/... or live URL, if available]*

PROBLEM 3

[one line — what was broken, slow, or missing]

Solution — *[what you built / the approach you took]*

Resources — *[team size, AI agents/tools, APIs, frameworks]*

Impact — *[time saved, revenue impact, users served, cost cut — real numbers]*

Live link — *[github.com/... or live URL, if available]*

CLOUD & INFRASTRUCTURE EXPERIENCE (25%)

List what you've actually used hands-on — not what you've read about.

- AWS: [services used — EC2, Lambda, S3, RDS, etc. + what you built with them]
- Google Cloud: [services used + what you built with them]
- Azure: [services used + what you built with them]
- Other: [Docker, Kubernetes, CI/CD pipelines, monitoring tools, etc.]

LEADING AI ADOPTION (SENIOR-SPECIFIC)

How did you personally bring AI-first practices into your team or company?

- [e.g., introduced AI code review tooling, trained the team on prompt/process workflows, built an internal AI agent for X]
- [Result — faster delivery, fewer bugs, adoption across teams, etc.]

SKILLS

Languages: [e.g., JavaScript, TypeScript, Python, SQL]

Frameworks & Libraries: [e.g., React, Next.js, Node.js, Express]

AI Tools & Platforms: [e.g., Claude, OpenAI API, LangChain, Cursor]

Cloud & DevOps: [e.g., AWS, GCP, Azure, Docker, CI/CD]

Databases: [e.g., PostgreSQL, MongoDB, DynamoDB]

EXPERIENCE (25%)

[Company Name] | [Your Role] | [Month Year] – [Month Year / Present]
[One to two lines: scope, team size, what you owned]

[Company Name] | [Your Role] | [Month Year] – [Month Year]
[One to two lines: scope, team size, what you owned]

[Company Name] | [Your Role] | [Month Year] – [Month Year]
[One to two lines: scope, team size, what you owned]

EXAMPLE — FILLED IN

Rohan Mehta

Senior Developer / Solution Architect

rohan.mehta@email.com | +91-98XXXXXX21 | linkedin.com/in/rohanmehta | github.com/rohanm

PROBLEMS SOLVED

PROBLEM 1

Finance team spent 15+ hours a week manually reconciling invoices across 3 disconnected accounting tools.

Solution — Built an AI agent that reads incoming invoices, cross-checks entries against the ERP, and flags mismatches automatically.

Resources — Solo build · Claude API · AWS Textract · Node.js backend

Impact — Reconciliation time dropped from 15 hrs/week to under 2. Saved the client roughly $18,000/year in ops cost.

Live link — Private repo — demo video on request

PROBLEM 2

A hospitality client's legacy booking system couldn't sync inventory in real time across 5 locations — causing double-bookings.

Solution — Re-architected the booking engine with an event-driven sync layer and a queue-based inventory service — plain systems design, no AI involved.

Resources — Team of 4 · AWS (Lambda, DynamoDB, EventBridge) · PostgreSQL

Impact — Eliminated double-bookings across all 5 locations. Booking-related support tickets dropped by 80%.

Live link — Case study available on request

PROBLEM 3

An ecommerce client's checkout page took 6-8 seconds to load on mobile, causing a high rate of cart abandonment.

Solution — Rebuilt the checkout flow with server-side rendering, lazy-loaded assets, and a leaner payment integration.

Resources — Solo · Next.js · Stripe API · Vercel Edge Functions

Impact — Load time cut to under 1.5 seconds. Mobile checkout conversion improved by 22%.

Live link — Private demo on request

CLOUD & INFRASTRUCTURE EXPERIENCE

- AWS: Lambda, DynamoDB, EventBridge, Textract, S3 — used in production across 3 client systems
- Google Cloud: BigQuery for client analytics dashboards, Cloud Run for containerized microservices
- Azure: Functions + Cognitive Services powering the support-ticket triage system
- Other: Docker, GitHub Actions CI/CD, Datadog for monitoring

LEADING AI ADOPTION

- Introduced a "prototype-before-proposal" workflow using AI — cut client scoping time roughly in half.
- Trained a 6-person dev team on structured AI-assisted development (process, not just prompting) — reduced rework cycles by ~30%.

SKILLS

Languages: JavaScript, TypeScript, Python, SQL

Frameworks & Libraries: React, Next.js, Node.js, Express

AI Tools & Platforms: Claude API, OpenAI API, AWS Textract, LangChain

Cloud & DevOps: AWS (Lambda, DynamoDB, EventBridge, S3), GCP (BigQuery, Cloud Run), Azure Functions, Docker, GitHub Actions

Databases: PostgreSQL, DynamoDB, MongoDB

EXPERIENCE

PiVisions Inc | Senior Developer / Solution Architect | Jan 2021 – Present
Lead custom app development for hospitality and ecommerce clients; manage a team of 5 across design, dev, and QA.

Nimbus Softworks | Software Engineer | Jun 2018 – Dec 2020
Built backend systems and payment gateway integrations for B2B SaaS clients.

[Junior Developer / Fresher — your actual title or area of interest]

email@example.com | +91-XXXXXXXXXX | linkedin.com/in/yourname | github.com/yourname | portfolio site

PROBLEMS SOLVED / PROJECTS BUILT (THIS IS 50% OF YOUR CV)

No professional experience yet is completely fine. Personal projects count — as long as they're real, live, and yours.

> **PROBLEM 1**
> *[one line — what was broken, slow, or missing]*
> **Solution —** *[what you built / the approach you took]*
> **Resources —** *[team size, AI agents/tools, APIs, frameworks]*
> **Impact —** *[time saved, revenue impact, users served, cost cut — real numbers]*
> **Live link —** *[github.com/... or live URL, if available]*

> **PROBLEM 2**
> *[one line — what was broken, slow, or missing]*
> **Solution —** *[what you built / the approach you took]*
> **Resources —** *[team size, AI agents/tools, APIs, frameworks]*
> **Impact —** *[time saved, revenue impact, users served, cost cut — real numbers]*
> **Live link —** *[github.com/... or live URL, if available]*

> **PROBLEM 3**
> *[one line — what was broken, slow, or missing]*
> **Solution —** *[what you built / the approach you took]*
> **Resources —** *[team size, AI agents/tools, APIs, frameworks]*
> **Impact —** *[time saved, revenue impact, users served, cost cut — real numbers]*
> **Live link —** *[github.com/... or live URL, if available]*

CLOUD & INFRASTRUCTURE EXPERIENCE (25%)

AWS, Google Cloud, and Azure all give free credits for exactly this reason — use them.

- AWS: [what you deployed, even a small project — EC2, Lambda, S3]
- Google Cloud: [what you deployed, even a small project]
- Azure: [what you deployed, even a small project]
- Deployment: [Vercel, Render, Netlify — link your live projects]

LIVE PROJECTS (ADDITIONAL 25%)

This is where freshers can stand out the most — don't skip it.

Project 1: *[live link — Vercel/Render/GitHub] — one line on what it does*
Project 2: *[live link — Vercel/Render/GitHub] — one line on what it does*
Project 3: *[live link — Vercel/GitHub] — one line on what it does*

SKILLS

Languages: [e.g., JavaScript, Python, SQL]
Frameworks & Libraries: [e.g., React, Next.js, Node.js]
AI Tools & Platforms: [e.g., Claude, OpenAI API, GitHub Copilot]
Cloud & Deployment: [e.g., AWS, Firebase, Vercel]

EDUCATION / BACKGROUND

[Degree / Course]: *[Institution] · [Year] — keep this short, one line*
Additional learning: *[courses, certifications, bootcamps — only if genuinely relevant]*

EXAMPLE — FILLED IN

Ananya Iyer

Junior Developer / Fresher

ananya.iyer@email.com | +91-97XXXXXX44 | linkedin.com/in/ananyaiyer | github.com/ananyaiyer

PROBLEMS SOLVED / PROJECTS BUILT

PROBLEM 1

Existing expense-tracking apps didn't handle splitting uneven group expenses well — a constant friction point with friends.

Solution — Built a group expense-splitter web app with custom split logic and AI-suggested categorization of each expense.

Resources — Solo · React · Node.js · OpenAI API for auto-categorization

Impact — 40+ real users across friend groups; consistently positive feedback on how accurate the AI categorization was.

Live link — expense-split.vercel.app

PROBLEM 2

Students on campus struggled to find used textbooks — scattered WhatsApp groups, no real marketplace.

Solution — Built a peer-to-peer textbook marketplace with search, filters, and in-app messaging between buyer and seller — no AI, just solid product thinking.

Resources — Solo · Next.js · Firebase · Cloudinary

Impact — 120+ listings posted in the first month on campus, entirely through word of mouth.

Live link — campusbooks.vercel.app

PROBLEM 3

Wanted hands-on understanding of AI agents, not just theory — needed a real, working use case to learn from.

Solution — Built a small AI agent that monitors a GitHub repo's activity and auto-drafts pull request summaries.

Resources — Solo · Claude API · GitHub Actions

Impact — In active use across 3 personal repos; noticeably cut down PR review prep time.

Live link — github.com/ananyaiyer/pr-summarizer

CLOUD & INFRASTRUCTURE EXPERIENCE

- AWS: Deployed a small Flask API on EC2; used S3 for storing uploaded book images
- Google Cloud: Hosted a static project site on Firebase Hosting
- Azure: Used free-tier Azure App Service for a college course project
- Deployment: All three live projects deployed and running on Vercel

LIVE PROJECTS

Project 1: *expense-split.vercel.app — group expense splitter with AI categorization*

Project 2: *campusbooks.vercel.app — peer-to-peer textbook marketplace*

Project 3: *github.com/ananyaiyer/pr-summarizer — AI agent for PR summaries*

SKILLS

Languages: JavaScript, Python, SQL

Frameworks & Libraries: React, Next.js, Node.js, Flask

AI Tools & Platforms: Claude API, OpenAI API (GPT-4 Vision)

Cloud & Deployment: AWS (EC2, S3), Firebase, Vercel

EDUCATION / BACKGROUND

B.Tech, Computer Science: *SRM Institute of Science and Technology · 2026*

Additional learning: *Self-guided course on building AI agents; participated in 2 campus hackathons*

Remember: there's no fixed method here. Adapt this to your own story — the point is to show what you've actually built and solved, not to follow a template word for word.