**Per Scholas Software Engineering Resume Playbook (ATS‑Ready)**

**Purpose & Outcomes**

Audience: Per Scholas Software Engineering learners in a 15‑week immersive.

Outcome: A tech‑centered resume that meets industry expectations and passes Applicant Tracking Systems (ATS).

Length: 1 page (entry/early career); 2 pages max if 5+ years relevant experience or multiple substantial projects.

**What “Good” Looks Like (Aligned to PD/JRA)**

HEADER

• Name (largest text), phone, professional email

• LinkedIn + GitHub URLs (custom slugs); City, ST (no full address)

PROFESSIONAL SUMMARY (3–4 lines)

• Target the role (e.g., Software Engineer | Front‑End Developer | Back‑End Developer)

• 2–3 strengths + 1–2 technologies + 1 outcome/impact + portfolio hint

TECHNICAL SKILLS

• Grouped, curated: Languages | Frameworks | Databases | Cloud/DevOps | Testing | Tools

• Prioritize skills from the job description; avoid laundry lists

PROJECTS (most important early career)

• 2–4 projects with Action + Tech + Impact bullets

• Include GitHub + live demo links + testing + CI/CD + role/ownership

EXPERIENCE

• Title, company, city, state, dates (MMM YYYY – MMM YYYY)

• 3–5 quantified bullets using real tech and outcomes

• Translate non‑tech roles to transferable skills; include tooling used

EDUCATION & CERTIFICATIONS

• Per Scholas Software Engineering (cohort/expected completion)

• Degrees, relevant coursework, honors; relevant certs

FORMATTING

• Single column; standard section headers; 10–12 pt body; consistent margins

• No tables/images/icons/multi‑column layouts (ATS‑risky)

PD MILESTONES

• Week 6 JRA: Name/TMAY/Resume – must be current and aligned to LinkedIn

• Week 10 JRA: LinkedIn + Professional Presentation + Job Hunting Spreadsheet + Interview Qs – resume and LinkedIn must match

**Recommended Section Order**

1) Name + Contact Links

2) Professional Summary

3) Technical Skills

4) Projects (then Experience if related, otherwise keep Projects above)

5) Experience (Professional/Related)

6) Education & Certifications

7) Awards | Activities | Volunteering (optional)

Tip: Keep section names standard (Projects, Experience, Education).

**Professional Summary – Formula & Examples**

Formula: Title + scope/strengths + key tech + impact + proof

GENERAL SE EXAMPLES

• Software Engineer specializing in React, Node.js, and PostgreSQL. Built and shipped full‑stack apps with REST/GraphQL, JWT auth, and CI/CD. Known for clean code, solid tests, and cross‑functional collaboration. Deployed 3 projects to cloud with monitoring.

• Full‑Stack Developer with TypeScript across front‑ and back‑end. Raised test coverage to 85% with Jest/Playwright; automated CI on GitHub Actions; delivered features across 6 sprints.

FRONT‑END EXAMPLES

• Front‑End Developer with expertise in React, TypeScript, and Redux. Implemented accessible UI components and cut bundle size 35% via code‑splitting and tree‑shaking. 98% Lighthouse accessibility on latest project.

• UI Engineer focused on performance, accessibility (WCAG 2.1 AA), and DX. Built component library with Storybook; improved p95 FCP by ~400ms through lazy loading.

BACK‑END EXAMPLES

• Back‑End Developer focused on Node.js, Express, and PostgreSQL. Designed resilient APIs, raised test coverage to 85%, and containerized services with Docker for consistent dev/prod parity.

• API Engineer with GraphQL/REST experience, Redis caching, and background jobs. Reduced 5xx errors by 60% after input validation and rate‑limiting refactor.

**Skills Section – Curated Templates**

GENERAL SE

• Languages: JavaScript (ES6+), TypeScript, Python, SQL

• Front‑End: React, Next.js, Redux Toolkit, Tailwind, Vite

• Back‑End: Node.js, Express, REST, GraphQL

• Databases: PostgreSQL, MongoDB, Prisma/Sequelize

• Cloud/DevOps: Docker, GitHub Actions, AWS (S3, CloudFront), Terraform

• Testing/Quality: Jest, Playwright, ESLint, Prettier

• Tools: Git, GitHub, Postman, Figma, Jira

FRONT‑END (prioritize UI/UX/performance)

• Languages: JavaScript (ES6+), TypeScript

• Libraries/Frameworks: React, Next.js, Redux Toolkit, Zustand

• Styling: CSS3, Sass, Tailwind, CSS Modules

• Testing: Jest, React Testing Library, Playwright

• Tooling: Vite, Webpack, Babel, ESLint, Prettier

• Accessibility: WCAG 2.1, ARIA, Axe

• CI/CD: GitHub Actions, Vercel

• Analytics/Monitoring: Lighthouse, Web Vitals

BACK‑END (prioritize data/API/reliability)

• Languages: TypeScript, JavaScript, Python

• Frameworks: Node.js, Express, Fastify, NestJS

• APIs: REST, GraphQL (Apollo)

• Databases: PostgreSQL, MySQL, MongoDB, Redis

• Auth/Security: JWT, OAuth2, RBAC, input validation (Zod/Joi)

• Messaging/Jobs: BullMQ, SQS, WebSockets

• DevOps: Docker, docker‑compose, Terraform

• Observability: OpenTelemetry, Winston, Grafana, Prometheus

• CI/CD: GitHub Actions, AWS (Lambda, ECS), Render

**Projects – What to Show & How**

Choose 2–4 projects that match the target role and demonstrate depth (auth, testing, CI/CD, performance, data).

Each entry should include:

• Name + 1‑line purpose (who/what problem)

• Stack (front‑end, back‑end, DB, cloud)

• 2–4 quantifiable bullets (Action + Tech + Impact)

• Links: GitHub repo(s), live demo

• Proof in README: build badges, coverage %, screenshots/GIFs, setup steps, test commands

Example (Entry‑Level, Full‑Stack)

TaskHero | React, Node.js, Express, PostgreSQL, Jest, Docker

Built a Kanban‑style task manager with drag‑and‑drop and role‑based access.

– Implemented JWT auth and RBAC; blocked unauthorized access across protected routes.

– Added Jest unit tests and Playwright E2E; 85% coverage and zero critical bugs at release.

– Configured GitHub Actions CI to run tests/build on PR; <6 min pipeline.

– Deployed with Docker to Render; uptime 99.9% over 30 days.

Example (Group Project, Front‑End‑heavy)

CareConnect | React, TypeScript, GraphQL, Apollo Client, Vite

Telehealth scheduling and messaging for clinics.

– Coordinated 5‑person team using Scrum; shipped MVP in 3 sprints.

– Implemented WebSockets for real‑time chat; p95 latency <150ms.

– Achieved 98% Lighthouse accessibility; added Axe checks to CI.

Example (Back‑End‑heavy)

LedgerLine API | Node.js, Fastify, PostgreSQL, Redis, BullMQ, Docker

Financial ledger with idempotent writes and background reconciliation jobs.

– Optimized PostgreSQL queries with indexes; reduced report time 9.2s → 1.1s.

– Introduced input validation + rate‑limiting; 5xx errors dropped 60%.

– Added background jobs with BullMQ; queues clear <1s under normal load.

**Bullet Formula + Copy‑Ready Bullets**

Formula: [Action Verb] + [What/How] + [Tech] + [Impact/Metric]

GENERAL / FULL‑STACK

• Built a full‑stack React/Node/Postgres app with role‑based auth; implemented CI/CD (GitHub Actions) to auto‑test and deploy to Render.

• Introduced TypeScript across codebase, removing implicit anys and catching 25+ type errors pre‑commit.

• Wrote Jest unit tests and Playwright E2E tests, raising coverage from 35% to 85%.

FRONT‑END

• Refactored state with Redux Toolkit + RTK Query, reducing re‑renders ~30% and improving API caching.

• Implemented code‑splitting and lazy loading, trimming initial bundle 35% and improving FCP ~400ms.

• Built accessible components with Headless UI and ARIA patterns, achieving WCAG 2.1 AA contrast.

BACK‑END

• Designed REST/GraphQL endpoints with Express/Apollo, adding rate‑limiting and validation; 5xx errors decreased 60%.

• Optimized PostgreSQL with indexes/EXPLAIN; heavy report improved 9.2s → 1.1s.

• Containerized services with Docker and docker‑compose; shortened onboarding by 2 days.

**ATS Do/Don’t**

DO

• Use standard headers (Summary, Skills, Projects, Experience, Education)

• Single column, plain bullets (•), black text, consistent dates (Jan 2025 – Aug 2025)

• Mirror keywords from the job description across Summary/Skills/Projects

DON’T

• No tables, text boxes, headers/footers with content, or icons/images

• Don’t keyword‑stuff tech you can’t discuss

• Don’t use multiple columns or fancy templates (parse risk)

**10‑Minute Tailoring Workflow (Per Application)**

1) Paste the job post into a doc; highlight must‑have skills and responsibilities.

2) Customize Summary to role + top 3–5 technologies from the post.

3) Reorder Skills so matching tools appear first; remove unrelated items.

4) Prioritize 1–2 most relevant projects; move Projects above Experience if needed.

5) Swap bullet verbs/tech to mirror JD language (e.g., “RESTful APIs”, “CI/CD”, “observability”).

6) Save as PDF with role‑specific file name (First\_Last\_Software\_Engineer.pdf).

**15‑Week Timeline (Mapped to PD/JRA)**

Weeks 1–2: Draft baseline resume; set LinkedIn & GitHub; start Project #1

Weeks 3–4: Add Project #1 bullets + links; curate Skills; refine Summary

Week 5: Peer review; run ATS checks; line‑edit for clarity

Week 6 (JRA Checkpoint): Present Name/TMAY/Resume; incorporate feedback

Weeks 7–8: Start Project #2; add tests; implement CI/CD; update bullets

Week 9: Quantify results (perf, coverage, users); refine Summary/Skills

Week 10 (JRA Checkpoint): LinkedIn + Presentation + Job Hunting Spreadsheet + Interview Qs; resume must match LinkedIn

Weeks 11–12: Tailor resume to 3 target roles; create role‑specific versions

Weeks 13–15: Apply with tailored resumes; maintain job tracker; iterate from recruiter feedback

**Recruiter Tips: What Tech Recruiters Say Works**

• Lead with relevance: The first third of your resume should answer “Why you for THIS role?” (title‑matched summary + matching skills + best project).

• Show proof fast: Link GitHub + live demos; pin your 1–2 best repos; add READMEs with setup, tests, screenshots, and a short GIF.

• Make bullets quantifiable: Perf (ms, %), coverage (%), reliability (error rates), usage (# users), iteration speed (sprints), team size.

• Prioritize depth over breadth: Fewer skills, more evidence (tests, CI, auth, performance work, data design).

• Keep it scannable: Clear headers, white space, consistent punctuation/tense; 4–6 bullets max per section, 1–2 lines per bullet.

• Title alignment: Match the job title on your resume summary to the posting (Software Engineer vs. Front‑End Developer), especially for early‑career screens.

• Keyword mapping: Mirror the JD’s exact nouns (e.g., “TypeScript”, “React”, “Jest”, “CI/CD”, “PostgreSQL”). Put the most important ones in Summary + top skills.

• Education matters, but projects win: Highlight 2–4 strong projects with clear outcomes. Bootcamp projects are fine—make them production‑like (auth, tests, CI/CD).

• Dates & gaps: Use MMM YYYY; if you have a gap, add “Independent Software Projects, YYYY–YYYY” with real output.

• File and links: Professional file name; working links; custom LinkedIn URL; clean commit history (sensible messages, PRs).

**Track‑Specific Tailoring: Front‑End vs Back‑End**

FRONT‑END FOCUS

• Keywords to emphasize: React, TypeScript, Redux/RTK Query, Next.js, CSS (Tailwind/Sass), accessibility (WCAG), Jest/RTL/Playwright, performance (FCP/LCP), Vite/Webpack.

• Project prompts: Component library with Storybook; a11y‑first redesign; dashboard with charts; PWA with offline caching; Next.js app with ISR.

• Interview‑ready bullets: a11y metrics, bundle size reduction, state mgmt improvements, real‑time UI (WebSockets), testing strategy.

BACK‑END FOCUS

• Keywords to emphasize: Node.js, Express/Fastify/NestJS, REST/GraphQL, PostgreSQL/MongoDB, Redis caching, background jobs (BullMQ), auth (JWT/OAuth/RBAC), Docker, CI/CD, observability.

• Project prompts: Financial ledger with idempotent writes; rate‑limited public API; ETL pipeline; async job queue; GraphQL gateway with federation.

• Interview‑ready bullets: query optimization, error‑rate reductions, latency p95/p99, data modeling choices, security hardening, scalability tests.

**Mini Templates (Copy‑Ready)**

HEADER

First Last · City, ST · (555) 555‑5555 · email@domain.com

LinkedIn: linkedin.com/in/handle · GitHub: github.com/handle

SUMMARY

Software Engineer skilled in React, TypeScript, Node.js, PostgreSQL, and CI/CD. Built and shipped full‑stack apps with JWT auth, testing (Jest/Playwright), and Docker. Passionate about readable code, accessibility, and measurable impact.

SKILLS (example – tailor per role)

Languages: JavaScript (ES6+), TypeScript, Python, SQL

Front‑End: React, Next.js, Redux Toolkit, Tailwind

Back‑End: Node.js, Express, REST, GraphQL

Databases: PostgreSQL, MongoDB

Cloud/DevOps: Docker, GitHub Actions, AWS (S3, CloudFront)

Testing: Jest, Playwright

Tools: Git, Postman, Figma

PROJECTS

DevMarket — Marketplace for dev tools

– Built Next.js app with API routes and Prisma/PostgreSQL; p95 <250ms on key endpoints.

– Added OAuth via NextAuth; zero auth regressions across 20+ tests.

– Deployed on Vercel with preview deployments per PR.

EXPERIENCE

Teaching Assistant, Per Scholas — City, ST | 2025

– Mentored 20 learners in JS/React, reviewing PRs and reducing bugs pre‑demo by 40%.

– Wrote example unit tests; improved coverage to 80% in demo repo.

EDUCATION

Per Scholas — Software Engineering (Immersive), Expected 2025

B.S. in [Major], University Name — City, ST | 2023

**Final Pre‑Submit Checklist**

□ Summary targets the exact role title

□ Skills grouped/curated; mirror JD keywords

□ 2–4 quantified project bullets each with tech + impact

□ GitHub & LinkedIn links present and professional

□ Single column; consistent spacing/fonts; no tables/images/icons

□ File name professional; PDF export verified

□ Education includes Per Scholas; dates consistent with LinkedIn

□ Ready for Week 6 and Week 10 PD checkpoints