Abhinav Yadav

Email: me@abhiyadav.in Mobile: +91 8630087242 Address: Noida, 201301, India
https://abhiyadav.in https://github.com/abhinav-yadav-official https://linkedin.com/in/abhiyd

OVERVIEW

I am the senior-most engineer at Instahyre (India-based recruitment startup) with *7* years of backend, full-stack, and infrastructure experience. I lead and mentor *20+* developers. I built the company's deployment platform, own the search infrastructure powering candidate-job matching, and run the review process that keeps risky changes from breaking production. The platform serves *5M+* users with *100M+* monthly API requests, *241M+* search documents, and *2B+* database rows.

TECHNICAL SKILLS

Languages:	Python, JavaScript, Go
Backend:	Django, Node.js, FastAPI
Databases:	MySQL, Redis (cache and queue)
Search:	Elasticsearch, OpenSearch, Painless scripting
Async / Messaging:	Celery, RabbitMQ
Frontend:	AngularJS, Next.js
Cloud / Infra:	AWS (EC2, RDS, S3, Lambda, CloudFront, Secrets Manager), Docker, Nginx, Linux
Observability:	Datadog (APM, dashboards), CloudWatch, Sentry

WORK EXPERIENCE

Software Development Engineer III, *Instahyre, Platform Architecture, Noida, India* Apr 2024 – Present

- The *Instamatch* candidates page **Elasticsearch** query used nested queries on the opportunities field, forcing per-candidate sub-document scans for basic filters. Flattened these into integer arrays on *241M* candidate documents and replaced the float salary range with a pre-scaled integer field. Candidate-fetch and count query both dropped *8*× to sub-*250ms*.
- Eliminated **Painless** scripts from the *Instamatch* candidates page query (critical opp API, *29M+* monthly requests), replacing with native **OpenSearch** term filters for skill, location, and hireable matching. Median latency fell *~11*× from *3s* to *260ms*.
- *Forge* is Instahyre's internal deployment platform, which I built from scratch to replace manual **SSH** deploys that consumed *~1* hour of engineering time daily. It handles guarded auto-merge, pre-flight schema and index checks, one-command rollback, and release dashboards. Deploy time dropped from *30* minutes to *4*, rollback MTTR from over *20* minutes to under *90* seconds, and a whole category of deploy-related P0 incidents stopped happening.
- Built *ESMultiWrite* to replace the one-off scripts written each time a search index needed rebuilding or migration. It makes alias swaps, parallel rebuilds, cross-version readers, and rollback paths explicit. Used it for the **ES 7** to **OpenSearch 3** cutover across *241M+* documents and *12+* production indices with zero search downtime; it's now the default pattern for any index work.
- The opportunity table was approaching its *32-bit* primary key ceiling with over *2B* rows on a live write path. Led the migration to **BIGINT**: dual-write to a shadow table, online backfill, and an atomic cutover that kept customer impact under one second. While profiling bulk inserts during the process, caught a deadlock pattern creating ID gaps that would have silently broken referential integrity. The approach became the template for later large-table migrations.
- Owned the P0/P1 incident review process across the backend org (*15+* engineers, *4* teams) after we kept seeing the same failure modes: schema changes without rollback plans, unchecked cache invalidation, deploys against untested indices. Built the taxonomy, runbooks, and review gate that surfaced risky changes pre-production. Incidents dropped *42%* over *18* months.
- Designed and shipped the 3-step guided job-posting flow that replaced the modal-based form. Recruiters now preview matching candidates, sidebar filters, and aggregate insights inline before alerts go out, giving them a chance to fine-tune the job for relevance. Rolled out behind per-employer flags with A/B enablement and a clean cutover for the rest.
- Built the Talent Insights panel that aggregates over the matching candidate pool (active + passive) and surfaces top companies, skills, schools, cities, and industries with counts and percentages. Surfaced inline during job posting and as a modal on the Candidates page; helps recruiters calibrate expectations before they spend outreach budget.

Senior Software Engineer, *Instahyre, Scale, Reliability & Modernization, Remote* May 2022 – Mar 2024

- The production monolith ran web serving and Celery workers on the same machines, so outbound processing bursts were competing directly with user-facing requests. Split it into three independent tiers: web serving, dedicated **Celery** worker pools, and the outbound pipeline, each with its own scaling and release cadence. Web-tier p99 tail latency dropped *35%* and the outbound team went from waiting on shared deploys to shipping roughly *3*× faster.
- Traced *Instamatch* production timeouts to their root causes. The worst was a conversation query fetching unread state for the employer's entire history instead of just the page of *30* candidates being displayed; scoping that brought it from *1.86s* to *38ms*. A filter query was taking *250s* because of a bad **MySQL** join order; a join hint fixed it at *290ms*. The filter-counts API (*6.83M* calls/month) was evaluating the same queryset *5* times per request; one query plus aggregation cut it *7*×. Large jobs still hit *30s+* timeouts; a **MySQL** covering index on the opportunity table resolved the remaining cases.

- Diagnosed silent matching corruption (recruiters seeing candidates with the wrong work-experience) caused by a race between an async opp-deactivation task and the candidate-apply API doing a full-row save over stale `is_active`. Scoped `opp.save` to `update_fields` only and reproduced the fix deterministically with a lock-based debug harness.
- Built a **Redis** cache-aside layer with per-key TTLs, event-driven invalidation, and pooled connections. Applied it to the highest-traffic endpoints first and got p95 latency down *38%*. Other teams adopted it; now in use across *20+* endpoints company-wide.
- Lifted the long-standing skill whitelist for direct employers. Built a default-mode inline skill picker on the Candidates page and a boolean mode that lets recruiters express AND/OR queries (e.g. `Java AND "Spring Boot"`) over the full *6.3K*-skill catalogue, mapped into the matching engine without polluting the default job form.
- Extended the team **RBAC** model to add Hiring Manager and Interviewer roles for Scheduler, scoped to own jobs/interviews, without breaking Admin/Recruiter semantics. Same model backs Scheduler, Reports, and Brand-page editing.
- Established the per-employer feature-flag rollout pattern. Every large feature now ships dark, dogfoods on the Instahyre employer, then rolls out to selected employers before broad release. Cut "first-day" incident clusters and made A/B testing default.
- Shipped admin Usage Reports (date-range + recruiter selection, async-generated and emailed) and the free-recruiter "request paid login" flow that emails admins from a pre-filled template; both surfaced upgrade paths previously chased manually.
- After a near-miss with a credential exposure, overhauled production access: least-privilege **SSH**, Unix user isolation, private network routing, and **AWS Secrets Manager** with automated credential rotation. Passed the next audit with zero findings.

Software Engineer, *Instahyre, Product Systems & Security, New Delhi, India* Aug 2019 – Apr 2022

- Built *Team Management* end-to-end: **RBAC** (Admin/Member roles), per-account license caps, audit trails for who added/deleted whom, deletion/restore with email notifications, and admin-only brand-page editing. Adopted by *1,000+* employer companies and *8,000+* recruiter seats; it's the core feature behind the premium pricing tier.
- Enforced premium-job credit limits (*250* profiles per *30*-day window) with atomic per-view decrement, auto-flip to inactive on credit exhaustion, and an expiry email. Closed a recruiter-support pain point and matched the pricing claim.
- Added Search filters for previous companies and undergrad/postgrad education ladders (e.g. B.Tech from a specific institute AND MBA from another). Pushed the new fields into the candidate index and sidebar UI without cluttering the default filter set.
- Cut mean incident detection time from around *18* minutes to under *9* by rolling out structured alerting, distributed tracing, and error-rate dashboards across the API and background processing tiers.
- Cut the *Instamatch* Kanban view load *2×* (*13.6s* to *6.3s*) by parallelizing stage requests and denormalizing three action-state fields onto the opportunity table to remove per-facet joins.
- Brought candidate onboarding down *2.5×* (*21s* to *8.6s*) via parallel backend calls and deferred **AngularJS** DOM per step.
- Replaced **Django** `get_or_create` in bulk opportunity inserts with `INSERT ...ON DUPLICATE KEY UPDATE`, cutting a *6k*-row batch from *91s* to *35s* and eliminating atomic-block lock contention.
- Fixed a recruiter-visible location-matching bug (NCR jobs surfacing candidates from the wrong cities). Rewrote the location predicate to handle the NCR/specific-location overlap correctly across the four `locations` × `candidate_locations` × `accept_outstation` cases and ran an opp-calc backfill across *3,780* affected production jobs.
- Built tooling for migration work the team kept doing by hand: a **Node.js** PDF-to-HTML conversion pipeline, **AngularJS** bulk refactor scripts, and **AWS Lambda** data-fix utilities. Prep time dropped from days to hours; reused across *6+* later projects.

Software Engineer Intern, *Fosterphi India, Greater Noida, India* Jul 2018 – Aug 2018

- Handwritten digit recognition web app: **Flask**, **Keras/TensorFlow**, **MNIST**, **JavaScript** canvas frontend.

OSS PROJECTS

LeetDrill, *LeetCode spaced-repetition system*

- Layers **SM-2** spaced repetition over your own submission history so weak topics resurface before you grind fresh problems; Chromium extension auto-captures every attempt.
- **Go + PostgreSQL** backend with an **HTMX** review UI — deliberately minimal stack so the scheduling algorithm is the only interesting part of the codebase.

Ichnos, *Crawler and search engine*

- Built from scratch (URL scheduling, crawl politeness, on-disk inverted index, **BM25** ranking) to understand how production search engines work end-to-end without off-the-shelf libraries.

Kleos, *Cold-outreach workflow engine*

- Sequencing pipeline with throttling, suppression lists, reply detection, and async workers.
- Per-recipient state and task-queue retries ensure one stalled send doesn't block the rest of the sequence, so a single bad address can't quietly stall a campaign.

EDUCATION

Dr. A.P.J. Abdul Kalam Technical University (AKTU), *Lucknow, India, B.Tech, Computer Science* 2015 – 2019