MEDEX WEBSITE PROJECT: Project Brief, Scope of Work & Functional Specification

MEDEX UPDATES

WEBSITE PROJECT

WWW.MEDEXJOB.COM

FOR:

PRAVIN KUMAR

SHAKIR PARVEJ

SUBMITTED TO :

WEB MAKING TEAM: MISS PRATYANCHA
MISS PRIYAL GUPTA

CO-ORDINATOR: MR. ARYAN GHEBAD

Dear Developers,

Welcome to **Medex** — a platform envisioned to become *the go-to ecosystem for India's medical professionals*.
This project isn't just about building another job portal — it's about creating a bridge between **talent and opportunity**, between **care and career**, across the entire spectrum of Indian healthcare.

As you work on this project, remember that every line of code you write, every table you normalize, every API you refine — will empower doctors, nurses, AYUSH practitioners, and paramedics across the nation to find meaningful work, serve communities, and build dignified careers.

Your work here contributes directly to **India's healthcare capacity** — and indirectly touches lives of millions of patients. That's a responsibility worth holding with pride.

So, as you code:

Write clean, secure, and human-readable code.

Build for scale, but design with empathy.

Keep doctors in mind — busy, overworked, and deserving of a seamless experience.

Remember that *clarity*, *stability*, and *trust* are as critical as innovation.

Always think of data privacy as **patient safety**, and performance as **care efficiency**.

The Medex vision is ambitious — to unify government, private, and international medical opportunities in one trusted, intelligent platform for Indian medicos.
With your talent and dedication, we'll make this not just a product, but a movement.

Thank you for being part of this journey.
May your code compile clean, your servers stay green, and your spirits stay high.
Let's build something India's healthcare community will truly be proud of.

— With gratitude and best wishes,
The Medex Core Team

MEDEX WEBSITE PROJECT

WWW.MEDEXJOB.COM

Opening summary:

Medex is a purpose-built medical jobs marketplace and recruitment platform focused on doctors, paramedical staff and AYUSH professionals. It combines a consumer-grade candidate experience with a robust, enterprise-ready HR offering and an admin/government channel for verified public jobs. The product includes three panels — **Candidate (user) panel**, **HR (employer) panel (HRs require admin approval and paid subscriptions), and an Admin panel (approves HRs, posts government jobs, moderates content)**.

This project specification set (25 sections + appendices) covers the full product lifecycle: product vision, personas, UX/SEO, frontend/backend architecture, data models, search and indexing, APIs, security & compliance (including HIPAA/GDPR/DPDP alignment), fraud & moderation, payments & subscription lifecycle, CI/CD, observations & runbooks, testing, SRE practices, roadmap, resourcing, pricing/monetization, marketing & growth, and operational appendices (SQL, ES mappings, email templates, glossary). It is written to serve as a single-source-of-truth for product, engineering, security, legal, finance and growth teams to design, build, launch and scale Medex.

INDEX

Executive & Product:

> HR panel: KYC, plans, post/manage jobs, view applicants, analytics, billing.

> Admin panel: HR approvals, govt jobs, news, moderation, payments oversight.

5. **End-to-end flows (detailed sequences)**

> Sequence diagrams & step-by-step flows for signup, KYC, job posting lifecycle, application lifecycle, payments, refunds, appeals, interviews and integrations (calendar, ATS).

Data, APIs & Search:

6. **Data model & schema notes (DDL-style examples)**

> Core tables: users, profiles, employers, jobs, applications, subscriptions, payments, audit_logs.

> DDL examples, indexes recommendations, denormalization notes and GDPR retention hooks.

7. **API design: endpoints, request/response examples & error codes**

> REST + optional GraphQL hybrid, auth conventions (JWT), endpoints for auth, profiles, jobs, applications, payments, admin.

> Pagination, sample payloads, standard error codes.

8. **Search architecture & indexing strategy**

> Hybrid approach: Elasticsearch/OpenSearch + semantic vectors (BERT/MiniLM).

> Index mappings for jobs/profiles/employers, pipeline (Kafka/RabbitMQ), embedding generation, ranking & personalization rules.

9. **Authentication, Authorization & RBAC matrix**

JWT + refresh, OAuth options, 2FA for privileged roles, RBAC & permission keys, middleware patterns, security best practices.

Payments, Trust & Security:

10. **Payments, billing & subscription lifecycle (detailed)**

> Pricing plans (1m,3m,6m,1y,2y,3y,5y,lifetime), payment gateway flow, webhooks, invoices, dunning & grace periods, refund policies, schema & endpoints.

11. **Verification, fraud detection & moderation rules**

> KYC flows, automated & manual verification, risk scoring heuristics, flagging rules, admin moderation UI & playbooks, appeals and audit trails.

12. **Security & compliance controls (detailed)**

> Encryption, IAM, WAF, OWASP mitigations, logging, SIEM, HIPAA/GDPR/DPDP mapping, DR/backup targets, testing & audits.

13. **Advanced Security & Compliance Architecture (expansion of Sec 12)**

> Defense-in-depth, SIEM/SOC design, pentesting cadence, compliance-as-code (OPA), certification roadmap (ISO/SOC2/HIPAA), vendor risk.

Privacy, Retention & Admin:

14. **Privacy & data retention policies**

> Data categories, consent management, user rights (access/rectify/erase/port), retention schedules, anonymization, breach notification SLA.

SCHEMA

Medex — Project Schema (complete system data model & DDL-style examples)

Below is a complete, practical project schema for **Medex** (Indian medical jobs portal). It encodes the product rules you described (Candidate / HR / Admin panels, HR approval & paid plans, admin-posted govt jobs, private jobs by HR, only admin-approved data goes to HR, etc.), covers candidates (doctors, AYUSH, nurses, paramedical), lists specialties & job formats, and provides production-ready DDL-style table definitions, indexes, lifecycle notes, and implementation guidance.

Use this as the canonical data model for engineering, product and infra teams.

---

1 — High-level overview / entities

Primary entities:

users — authentication record for everyone (candidates, HR users, admins).

candidate_profiles — extended profile for job seekers (medical/AYUSH/paramedical).

employers — hospitals, clinics, labs, institutions.

hr_accounts — HR users tied to employers (must be admin-approved).

kyc_documents — KYC uploads for employers/HR.

plans / subscriptions / payments / invoices — billing & monetization.

jobs — all job postings (private + govt), status controlled by admin approval and HR subscription.

applications — candidate applications to jobs.

resumes / documents — uploaded CVs, certificates.

audit_logs / admin_actions — immutable logs for compliance.

news / govt_jobs_feed — admin-managed announcements & govt jobs (govt jobs stored in jobs table with is_government flag).

specialties and job_formats — canonical lists (specialties = medical fields; job_formats = duty doctor, RMO, etc.).

search_index_meta — optional denormalized/search-trigger metadata.

Relationships:

users 1→1 candidate_profiles (if candidate)

employers 1→N hr_accounts

employers 1→N jobs

hr_accounts 1→N jobs (created_by_hr_id)

admin approves hr_accounts and jobs

15. **UX, accessibility & SEO specifications**

> Mobile-first design system, WCAG 2.1 AA rules, component-level accessibility, SSR/ISR Next.js SEO patterns, JSON-LD job schema, Core Web Vitals targets.

16. **Performance, scalability & reliability**

> SLOs, capacity planning, stateless architectures, DB patterns, search hot/warm, caching, autoscaling, load-testing scenarios, chaos testing, runbook checklist.

17. **CI/CD, staging & deployment strategy**

> Git branching and artifact promotion, IaC + GitOps (Terraform/ArgoCD), canary/blue-green, migrations, secrets management, security gates in pipeline.

18. Monitoring, logging & incident response

> Observability stack (Prometheus/Grafana, ELK, OpenTelemetry), SLIs, alerts, incident roles, runbooks (payments, search, breaches), post-mortem process.

19. **Testing strategy & acceptance criteria**

> Unit → integration → contract → E2E → perf → security → accessibility testing matrix, CI gating, sample Gherkin acceptance tests and release-blocking criteria.

Release, Team & Business:

20. **Release roadmap & milestones**

> Phases: Discovery → Alpha (MVP) → Beta → Launch → Post-launch; go/no-go checklist, KPIs per milestone, rollout & rollback rules.

21. Team & resourcing estimates

> Recommended org chart, hires by phase, person-month estimates, contractor vs FTE guidance, onboarding and ramp plans, hiring priority.

22. **Pricing & monetization strategy**

> Pricing tiers, pay-per-post, boosts, enterprise & lifetime plans, billing rules, tax & compliance, revenue KPIs, monetization experiments.

23. **Marketing, SEO & growth recommendations**

> GTM: content-led SEO, ABM for hospitals, LinkedIn & Google ads, referrals, partnerships (medical colleges/state health), 90-day launch plan and 12-month growth roadmap.

25> **Appendices: email templates, sample SQL indexes, sample search mappings, glossary**

Transactional email templates, production-safe Postgres index scripts & guidance, Elasticsearch index settings and mappings, JSON-LD examples, full glossary.

Appendices & Deliverables

Ready-to-use artifacts available:
• API spec (REST + GraphQL examples)
• DDL snippets + migration templates
• Postgres index SQL (CONCURRENTLY patterns)
• Elasticsearch mapping JSONs + hybrid query examples
• CI/CD skeleton (GitHub Actions example)
• Sample runbooks & incident templates
• Email templates (HR, candidate, billing, appeals)
• Gherkin acceptance scenarios and E2E examples
• k6 load-test scripts, Grafana dashboards, Kubernetes HPA manifests (available on request)

1. Executive summary

Medex is a vertical job marketplace that connects verified medical professionals (doctors, nurses, paramedics and AYUSH practitioners) with trusted healthcare employers — hospitals, clinics, labs and government bodies — by combining strict verification, subscription-based employer access, advanced matching/search, and a curated government-jobs stream.

Mission & vision:

Mission: Build the most trusted, high-signal hiring marketplace for healthcare — dramatically reducing time-to-fill for employers while protecting candidate privacy and professional credentials.

Vision: To be the default career platform for healthcare talent globally: where verified medical professionals find the best roles and institutions reliably find verified hires.

One-sentence value propositions:

For candidates: a single, trusted source of medical jobs with verification, confidential search, and personalized matching.

For employers/HR: a verified talent pool, compliance-ready employer onboarding, and measurable ROI on hires via subscription and analytics.

For platform/admin: predictable, recurring revenue from subscriptions and upsells, low fraud exposure because of admin-moderated govt jobs & KYC.

Problem statements (what we solve):

Many job sites flood candidates with low-signal or fraudulent postings and unverified employers.

Healthcare hiring requires credential verification, shift/roster specifics, and nuanced role definitions (visiting consultants, locums) that general job boards don't capture.

Employers expend time and money on irrelevant applicants; HR needs faster, higher-quality screening and scheduling tools.

Government healthcare vacancies are fragmented and often poorly distributed to relevant clinical audiences

---

Core solution (how Medex solves it):

Two distinct sections: **Private jobs** (employer/HR posted, subscription-gated) and **Govt jobs** (admin-posted, authoritative and highlighted).

Mandatory employer KYC and optional candidate license verification with visible badges for trust.

Advanced, medical-aware search/matching (specialty normalization, experience, shift types, geolocation radius).

Subscription plans for HRs with posting quotas, featured listings and candidate-pool access to monetize consistently.

End-to-end hiring workflows: post → match → message → schedule → offer → hire.

Target users & stakeholders:

Primary users :

> Specialist and general physicians (MBBS, MD, MS), consultants and residents.

> Nurses, allied health professionals, paramedic staff.

> AYUSH practitioners (Ayurveda, Homeopathy, Unani, etc.).

Secondary users

> Hospital HR teams, private clinic HRs, recruitment agencies specializing in healthcare.

> Government hiring committees and public health departments.

> Medical colleges and training institutions (source of early-career talent).

Stakeholders

> Founders & investors (revenue, retention).

> Platform operations & trust teams (fraud prevention).

> Legal/compliance (data privacy & employment law).

> Marketing (acquisition & partnerships).

Business & monetization model:

Primary: recurring subscription plans for HR/employers (tiered by duration and features: posting quota, featured slots, candidate pool access).

Secondary: pay-per-feature (featured/urgent listings, email blasts), enterprise contracts (API/ATS integration, invoicing), talent pool access, sponsored content & training partnerships.

Optional future: transactional fees for placements, background-check upsells, certification/CME marketplace.

Key performance indicators (KPIs) — platform-level

Pick initial targets for MVP then iterate. Example KPIs to measure success:

Acquisition: new candidate signups/week; new HR signups/week.

Activation: % candidates with completed profile (goal: 40–60% in early months).

Engagement: average applications per job; DAU/MAU.

Conversion/Revenue: MRR from HR subscriptions; employer conversion rate (signup → paid).

Quality: % of candidates & employers verified; % jobs leading to interviews/offers (tracked via HR reporting).

Operational: search latency (<200ms median), platform uptime (≥99.9%).

Strategic objectives & sample OKRs

Objective 1 — Launch a high-trust MVP

KR1.1: Onboard 2000 verified candidate profiles.

KR1.2: Sign up 100 HR accounts and convert 25% to paid within 30 days.

KR1.3: Admin-posted govt jobs publish and receive 5k unique views in first month.

Objective 2 — Demonstrate hiring ROI for HRs

KR2.1: Average time-to-first-qualified-candidate per job <14 days.

KR2.2: 10% of paid HRs report a successful hire within 60 days.

Objective 3 — Platform reliability & trust

KR3.1: Zero major security incidents; complete initial pentest before launch.

KR3.2: 95% of employer signups pass KYC in under 3 days.

Competitive positioning & differentiation:

Vertical focus on healthcare (vs. generalist job boards) enabling domain-specific features (medical registration verification, shift patterns, visiting consultant roles).

Admin-moderated govt jobs provide an authoritative stream absent from most private marketplaces.

Subscription-first employer model reduces low-quality urgent listings and encourages long-term engagement and predictable revenue.

Trust & compliance emphasis — KYC, verification badges, and fraud detection designed for the sensitivity of medical hiring.

Market assumptions & sizing (high level)

There is a large and continuously replenished pool of healthcare professionals (students, residents, nurses, allied health), and a recurring hiring need from hospitals, clinics, and public sector health departments.

Niches (specialist consultants, locum/visiting roles, AYUSH) are underserved by mainstream job boards.

Key risks & mitigations

Risk: Fraudulent employers or fake listings → **Mitigation:** KYC + admin review + heuristics + user reporting/appeals.

Risk: Low employer willingness to pay early → **Mitigation:** pilot enterprise accounts, provide trial credits, focus on high-value hospitals & recruiters.

Risk: Candidate privacy concerns → **Mitigation:** anonymous apply, granular privacy controls, clear privacy policy.

Risk: Heavy traffic spikes at govt-job announcements → **Mitigation:** caching, pre-warming, queued notifications, staggered publishing.

Scope boundaries for MVP:

In scope (MVP):

Candidate sign-up/profile/resume upload.

Job posting by HR (post-approval + paid subscription enforcement).

Admin-posted govt job stream & basic CMS.

Search & filtering, apply flow, email notifications.

Basic HR dashboard (post/manage jobs) + admin KYC queue.

Payment checkout + webhook handling for subscriptions.

Out of scope (initial MVP):

Full ATS integrations, advanced ML matching, background-check integrations, multi-region legal compliance beyond target launch country (unless prioritized), enterprise SSO (unless contracted).

Quick go-to-market highlights:

Priority acquisition channels: direct outreach to hospital HRs, partnerships with medical colleges, targeted LinkedIn campaigns for recruiters, content hubs for job-seekers (salary guides, interview prep).

Trust-building: highlight verification badges, curate gov-job listings, publish case studies from pilot hospital hires.

Launch offers: discounted enterprise onboarding, featured-listing credits for early adopters.

Success criteria (what “good” looks like at 6 months)

Platform has active user base: at least **10k candidates** created, **200 paying HR accounts**, and **>5,000 job applications** processed.

Revenue: positive MRR growth month-over-month with employer churn < 8% monthly.

Operational: system stable with >99.9% uptime, search median latency <200ms, and no unresolved major fraud incidents.

Immediate next steps (actionable)

Validate top assumptions (willingness-to-pay, verification feasibility) via 10–15 discovery interviews with HRs and 20 candidate interviews.

Prioritize MVP feature list into a 6–12 week sprint plan (I can produce this next).

Lock payment gateway options for the target country and set up sandbox accounts.

Prepare KYC checklist & admin review SOPs.

Build a single-page launch landing with an email capture and “Apply for early access” flows for HRs and candidates.

---

2. Expanded product vision & value propositions

Below is a richly detailed expansion of Medex's product vision and concrete value propositions — framed so product, sales, and engineering teams can directly translate them into features, go-to-market messaging, and prioritization.

Vision statement:

Medex will be the global, trust-first hiring platform for healthcare — the place where verified indian medical professionals and verified healthcare employers meet, transact, and build careers with privacy, speed, and compliance.

Product mission:

Enable faster, safer, and higher-quality healthcare hiring by combining domain-specific job discovery, rigorous verification, subscription-driven employer engagement, and workflows tuned to the cadence of clinical hiring.

Core value pillars (what the product must always deliver)

Trust & credibility — Verified employers and verified professionals, clear provenance of every listing.

Relevance & signal — Fewer false positives; better candidate-job fit through medical-aware taxonomies and matching.

Efficiency — Reduce time-to-hire via workflow tooling (apply → schedule → hire) and analytics.

Privacy & safety — Candidate-controlled contact details and admin moderation for sensitive hiring.

Predictability — Subscription revenue and predictable employer experience (quota, featured slots, reporting).

Primary value propositions (audience-specific)

For Candidates (doctors, nurses, paramedical, AYUSH):

High-signal job discovery: only medical/healthcare roles with structured data (specialty, shift type, bed count, on-call).

Verified employers: less risk of fraudulent or misleading postings.

Confidential search: anonymous apply and privacy controls for active/ passive seekers.

Career toolkit: resume builder, license registry, interview prep content, salary benchmarks, CME links.

Faster outcomes: streamlined apply + calendar-integrated interview scheduling shortens the hiring loop.

Messaging example: “Find verified medical jobs — confidentially — and get matched to roles that respect your specialty and schedule.”

For HR / Recruiters (hospitals, clinics, labs):

Higher quality pipeline: verified candidates, parsed resumes and auto-match scoring reduce screening overhead.

Predictable costs: subscription plans instead of unpredictable per-post fees; budgetable employer spend.

Operational tools: ATS-like pipeline, interview scheduling, team roles/permissions, analytics on time-to-fill & application quality.

Employer branding: featured listings, company pages, and sponsored content to attract passive candidates.

Compliance-ready workflows: KYC, document management, and audit logs for regulated hiring.

Messaging example: “Cut screening time and hire the right clinicians faster — with verified candidates and recruitment tools built for hospitals.”

For Admin / Platform Owners:

Monetizable inventory: recurring subscriptions and add-on features (featured slots, candidate pool access).

Low fraud exposure: moderated govt jobs + KYC reduce abuse and increase trust.

Network effects: verified hires & employer advocacy attract more users and reduce churn.

Scalable operations: subscription model reduces pressure on transactional collections and simplifies LTV planning.

Messaging example: “Build a trusted, repeatable revenue engine while protecting the clinical hiring ecosystem.”

Key product capabilities mapped to value propositions

Verification & badges → Trust & credibility for candidates and HR.

Medical taxonomy & specialty normalization → Relevance & better matching.

Advanced search & geo-radius → Faster discovery for candidates and targeted sourcing for HR.

Subscription management + quotas → Predictability for finance and HR.

Resume parsing & candidate scoring → Efficiency for screening.

Anonymous apply & privacy toggles → Candidate privacy and passive search safety.

Interview scheduler & calendar sync → Operational efficiency and candidate experience.

Admin CMS for govt jobs → Unique, authoritative content attracting traffic and trust.

Analytics & dashboards → Measure ROI for HR and platform KPIs for admins.

Usage scenarios (concrete examples)

1.Resident seeking fellowship: filters by specialty, sees verified tertiary hospitals offering fellowships, applies anonymously, schedules interview — hired within 6 weeks.

2.Private hospital HR hiring 3 nurses quickly: posts job under subscription, uses candidate pool search + auto-match, shortlists 10 within 48 hours, fills roles in 10 days.

3.State health department posts govt vacancies: admin posts vetted govt jobs reaching thousands of active candidates with a single announcement; peak traffic handled by caching & queuing.

Competitive differentiators (how Medex stands out)

Domain verticalization: everything about jobs (shifts, licensure, visiting roles) is medical-native.

Admin-moderated govt stream: authoritative government roles appear reliably and build trust — uncommon on private boards.

Subscription-first employer model: aligns long-term incentives, discourages frivolous postings.

Privacy-forward candidate flows: built-in anonymous apply & control over who sees sensitive info.

Verification-first marketplace: badges backed by KYC/medical registration reduce noise and litigation risk.

Monetization alignment — connect features to revenue:

Subscriptions (core): monthly/quarterly/year/lifetime — tied to job quotas and seat counts.

Featured & promoted placements (upsell): charged per slot/day — increases visibility for critical hires.

Talent pool access: pay-per-search or seat-based access to parsed CV DB.

Enterprise integration & invoicing: higher-tier recurring contracts (API/ATS).

Ads & sponsored content: CME providers, training partners, device vendors (carefully regulated).

Product roadmap (how vision maps to phases):

Phase 1 (MVP): Verified employers + admin govt jobs, candidate profiles, search & apply, subscription billing.

Phase 2 (growth): Resume parsing, candidate scoring, in-app messaging, interview scheduling, featured listings.

Phase 3 (scale): Enterprise ATS integrations, background checks, ML matching & recommendation engine, multi-region rollout.

Each phase preserves the core pillars: trust, relevance, efficiency.

Metrics that prove the vision (what to measure):

Engagement: % candidates with completed profiles; average applications per candidate.

Conversion: HR signup → paid subscription conversion rate.

Quality: % verified profiles; candidate-to-interview conversion rate.

Speed: median time-to-first-qualified-candidate; median time-to-hire.

Revenue: MRR, ARPU, churn.

Go-to-market positioning (short taglines)

“Medex — Verified medical jobs, faster hires.”

“Hire trusted clinicians. Protect patient care.”

“Confidential search for doctors. Predictable hiring for hospitals.”

Partnerships & integrations (strategic opportunities)

Medical councils / licensing bodies: automated verification (where available).

Medical colleges & residency programs: pipeline for early-career talent.

Payment providers: region-appropriate billing and enterprise invoicing.

Background-check providers & HR tech vendors: upsellable integrations for enterprises.

Job distribution partners: syndication to niche healthcare publications.

Risks specific to product vision & mitigations:

Low employer adoption of subscriptions → pilot program with enterprise hospitals, offer time-limited credits and white-glove onboarding.

Coverage limited by regional regulations → localize payment, legal T&Cs, and KYC expectations; stagger regional rollouts.

Fake verification attempts → multi-factor verification: domain/phone checks + manual KYC review + audit logs.

Brand & tone guidance (how to present Medex):

Tone: authoritative, empathetic, clinical-professional.

Visual cues: clean medical palette (white, navy, teal), clear typography, trust seals/badges prominent on profiles and employer pages.

Content voice: use data and procedure language for HR; career-oriented, empathetic voice for candidates.

What success looks like under this vision:

A trusted, repeatable employer acquisition funnel (enterprise and mid-market hospitals) with low churn; candidate uptake across specialties; govt job partnerships creating recurring spikes in traffic and long-term trust; measurable time-to-hire improvements published via case studies.

---

3. Personas & user journeys

This section builds concrete, product-ready personas and step-by-step user journeys for Medex. Use these to drive UX flows, acceptance criteria, analytics events, and sprint stories. For each persona I provide: at-a-glance profile, goals & success metrics, pain points, prioritized feature set, detailed happy-path journey with alternate/error branches, required data & permissions, and example screens/notifications. At the end I include a compact mapping of persona → top features and a job-to-hire funnel with metrics to instrument.

Persona 1 — Dr. Pravin (Specialist physician — mid-career)

Summary:

Age: 34. Urban. MD Cardiology. 6–10 years experience. Actively looking for consultant/associate professor roles. Tech comfortable but values privacy.

Frequency: visits portal weekly; applies selectively.

Primary goals

Find senior consultant roles that match specialty, sub-specialty and shift preferences.

Maintain confidentiality until shortlisted.

Rapid scheduling & interviewing without long email chains.

Success metrics (for product)

% relevant job matches clicked.

Conversion: saved search → application.

Time from application → interview invite.

Pain points

Low-signal job boards with unrelated posts.

Employers contacting current employer (privacy risk).

Manual interview scheduling.

Top features for Dr. Pravin

Filtered search by specialty/subspecialty and shift.

Anonymous (masked) apply option.

License/medical registration verification badge.

Calendar-integrated interview scheduling.

Job alerts & recommended jobs.

Happy-path user journey (detailed steps)

Discovery & signup

Visits Medex via organic search or campaign.

Sees landing copy focused on verified medical jobs and confidentiality.

Signs up with email + phone or LinkedIn import. Verifies phone via OTP.

Guided onboarding prompts to add registration number, specialty and experience.

(Event to track: `signup.complete`, `profile.start`)

Profile completion

Uploads CV (PDF) — resume parsing extracts education, experience, specialties.

Adds medical council registration number and uploads license image for verification.

Sets privacy preference: `public`, `mask-contact`, or `private`.

(Event: `profile.complete`, `license.submitted`)

Search & discover

Enters “cardiology consultant” + city filter. Uses radius slider (50 km).

Applies experience & salary filters; sorts by relevance.

Saves search and enables job alert (daily/real-time).

(Event: `job.search`, `search.save`)

Apply

Opens a job detail. Employer is “verified” (badge visible).

Chooses “Apply — Masked” to hide name/email; platform forwards application.

Uploads cover letter and submits one-click. Confirmation screen and email sent.

(Event: `application.submitted`)

Interview

HR shortlists her; uses Medex scheduler to propose time slots.

Dr. accepts a slot; a calendar invite with video link is generated.

(Event: `interview.scheduled`, `notification.sent`)

Offer

HR uploads offer letter PDF; Dr. views and accepts within portal or requests negotiation.

After acceptance, she updates status to “hired” and optionally marks job as filled publicly.

(Event: `offer.accepted`)

Alternate flows & error handling

License verification fails / flagged: Status set to `pending` and admin requests secondary proof (ID + council lookup); candidate receives email with next steps.

Apply limit reached (free account / quota): Show upsell prompt to upgrade account (if applicable) or suggest contact HR.

Interview conflicts: Provide reschedule flows and suggested alternative slots.

Data captured & permissions

Personal info, CV, license images (encrypted), privacy settings.

Permission: read/write to profile, send notifications, read calendar (if calendar sync enabled only after OAuth consent).

Example screens & notifications

Profile Builder (wizard)

Job Results with specialty tags + verification badge

Job Detail (Apply CTA: “Apply (Masked)”)

Application confirmation modal + email

Interview Invite modal + calendar attachment

Persona 2 — Rahul (Nurse — early-career)

Summary:

Age: 24. Recently graduated nurse. Mobile-first user, seeks nearby hospital jobs with day shifts and training opportunities. Visits multiple times per week.

Primary goals

Quickly find suitable entry-level nurse positions near his city.

Apply with minimum friction (mobile-first).

Pain points

Complex forms and heavy CV upload steps.

Lack of clarity about shift timings and salary.

Top features

Mobile-optimized search & apply.

Quick-apply with basic profile + one-click resume upload from phone.

Shift filter and clear salary/benefits display.

Saved jobs & push notifications.

Happy-path journey

Installs PWA or uses mobile web.

Quick sign-up via phone OTP.

Fills short profile (education, city, preferred shifts).

Searches: “staff nurse” + city; filters: day shift only.

Uses “Quick Apply” which attaches parsed resume and short answers.

Receives push notification when application changes status.

Alternate flows

Slow mobile upload: Allow resume via WhatsApp/email fallback or link to Google Drive.

No internet during apply: Save draft locally and sync when online.

Data & permissions

Phone number required. Push notification opt-in requested. Optionally location (approx) for radius search.

Persona 3 — Suresh (AYUSH Practitioner — private clinic owner / job seeker)

Summary:

Age: 45. Practicing Ayurveda. Looking for locum consultant roles and part-time teaching opportunities. Comfortable with basic web use.

Primary goals

Find locum or part-time positions and maintain a professional profile with certifications.

Pain points

Lack of proper categorization for AYUSH roles on mainstream boards.

Missing trust signals for employers seeking AYUSH practitioners.

Top features

AYUSH-specific categories and filters.

Certification upload & badge system.

Part-time / locum / visiting consultant job types.

Journey highlights

Onboarding with specialty selection (Ayurveda → Panchakarma).

Uses “Availability Calendar” to mark days available for locum work.

Receives SMS/Email when HR posts locum needs matching availability.

Persona 4 — Priya (Hospital HR Manager)

Summary:

Age: 32. HR manager at a 200-bed private hospital. Responsible for hiring doctors, nurses and lab technicians. Uses desktop majority but mobile for urgent tasks.

Primary goals

Source qualified candidates quickly; reduce time-for-screening.

Keep hiring within budget and measure ROI on postings.

Pain points

High volume of irrelevant applicants.

Manual scheduling and multiple email threads.

Top features

Employer dashboard with job quotas & subscription status.

Candidate pipeline (Kanban) with tags, notes, and stage-based workflow.

Advanced candidate search + auto-match suggestions.

Interview scheduling & bulk communication templates.

Invoicing & subscription management.

Happy-path journey

Registers employer account and uploads KYC documents. Admin approves within SLA.

Chooses subscription plan and makes payment via checkout. Subscriptions activate posting quota.

Creates job using Job Builder (structured fields + required documents).

Job is published; auto-match suggests top 10 candidates with match scores.

HR shortlists, sends batch interview invites, syncs with Google Calendar.

Marks candidates as hired/rejected with notes; downloads candidate list for HR records.

Alternate flows & errors

KYC rejection: HR receives rejection reason and can re-submit docs.

Payment failed: system retries, and HR receives grace period with limited posting until resolved.

Data & permissions

Employer admin can invite teammates and assign roles (owner, recruiter, viewer). RBAC applies. Data: company KYC, billing, posted jobs, candidates for company only.

Screens & notifications

HR Dashboard (quota, active jobs)

Job Builder (preview + SEO meta)

Candidate Pipeline (drag & drop)

Payment & billing center (invoices)

Persona 5 — Aryan (Recruiter / Agency)

Summary:

Third-party recruiter who posts on behalf of multiple hospitals. Needs multi-account support and API access for enterprise customers.

Primary goals

Manage multiple employer accounts at scale and get resumes fast for roles across cities.

Pain points

Per-employer quotas and seat limitations.

Repetitive posting workflows.

Top features

Multi-company management or sub-accounts.

Bulk job upload (CSV).

API/ATS integration (export & webhook).

Candidate pool subscriptions & credits.

Journey notes

Registers as agency, links multiple employer profiles, pays per employer or enters enterprise contract. Uses CSV bulk upload to create hundreds of similar job postings and uses candidate pool search to source resumes quickly.

Persona 6 — Shakir (Platform Admin / Moderator)

Summary:

Responsible for employer verification, managing govt job postings, handling fraud reports, platform health metrics, and legal requests.

Primary goals

Keep platform trusted by ensuring KYC, moderating suspicious content, and publishing govt jobs reliably.

Pain points

High manual load for KYC if not well automated.

Handling appeals, disputes and urgent fraud cases.

Top features

Admin KYC queue with document viewer and decision log.

Gov job CMS with scheduling & pinning to top.

Fraud dashboard (flags, heuristics, actions: suspend, warn, ban).

Audit logs and reports.

Admin journey

Reviews HR KYC documents; approves or rejects with notes. Receives alerts on suspicious patterns (many jobs posted from same IP) and can pause employer accounts. Schedules govt job release with embargo and caching pre-warm to handle traffic spikes.

Persona 7 — **Priyal (Govt Hiring Officer)**

Summary:

Publishes official government health posts. Needs authoritative, timely reach to candidates and wants to ensure authenticity of posted jobs.

Primary goals

Publish vacancies with clear criteria and timelines. Ensure wide reach to candidates and minimal disputes.

Pain points

Bureaucratic delays in content publishing; needs scheduled publishing & embargo support.

Top features

Government-job publishing portal (admin-only).

Scheduled publishing & prominent placement.

Dedicated govt-job notifications and feeds.

End-to-end user journeys (cross-persona scenarios)

Below are a few complete sequences that combine personas to highlight multi-party flows.

Scenario A — Private job posted → hire (HR Priya & Dr. Pravin)

Priya (HR) creates job and publishes after subscription quota validated.

Job appears in search and is sent in daily digest to matching candidates.

Dr. Pravin applies (masked). Priya receives application in pipeline, shortlists.

Priya offers interview slots → Dr. accepts → interview → offer → hire.
Metrics to instrument: job.view, application.rate, time-to-first-interview, offer.rate.

Scenario B — Govt job announcement (Govt Officer & many candidates)

Priyal uploads govt job via admin portal, schedules publish for 10 AM.

Admin pre-warms cache and pings notification queue. At 10 AM, job goes live and targeted alerts are sent.

Candidates apply; admin may publish official clarifications in news section.
Metrics: peak concurrent users, notification delivery rate, application volume per hour.

Scenario C — Fraud detection & intervention (Admin & HR / Candidate)

System flags employer for suspicious behavior (phone mismatch, domain-less email). Admin reviews KYC and suspends account.

Candidates who interacted with flagged employer receive notification and remediation instructions.
Metrics: flagged accounts, time-to-resolution, false positive rate.

UX & interaction patterns per persona

Mobile-first for Rahul & early-career candidates: simplified forms, image-based uploads, OTP login.

Desktop-optimized for HR Priya & recruiters: data tables, bulk actions, advanced filters, CSV export.

Admin heavy interface: document viewer, action buttons (approve/reject), audit trails.

Privacy & permissions map (what each persona can see/do)

Candidate: manage own profile; see job listings; see employer basic info; apply; manage privacy settings.

HR (employer): create & manage jobs for their company; view candidates who applied to their jobs; invite teammates; view billing/invoices.

Recruiter/Agency: access multi-company where authorized.

Admin: full platform access (approve HR, manage govt jobs); can view but must comply with privacy handling when viewing candidate PII (audit log).

Acceptance criteria examples (derived from journeys)

Candidate can complete profile and upload a CV; parsed data populates profile fields (at least 70% of expected fields correctly parsed for standard CV).

Candidate can apply to a job with masked identity option; HR receives application with masked contact info and can request full contact on shortlist.

HR post + payment flow: after successful payment webhook, HR gets posting quota and can publish up to quota; invoices generated and downloadable.

Admin can approve or reject employer within the KYC queue and the decision recorded in audit logs.

Analytics events to instrument (per journey stage — shortlist)

Use consistent event names and properties to support funnel analysis:

`signup.start`, `signup.complete`

`profile.update` (fields changed)

`license.submitted`, `license.verified`

`job.search` (query, filters, location, radius)

`job.view` (job_id, employer_id)

`job.save`, `job.apply` (apply_type: masked|open, resume_id)

`application.status_change` (status, actor_id)

`interview.scheduled` (slot, calendar_integration)

`payment.checkout_started`, `payment.success`, `payment.failed`

`employer.kyc_submitted`, `employer.kyc_approved`, `employer.kyc_rejected`

Persona → Top 5 features (quick mapping)

Persona	Top 5 Features
Dr. Pravin (Specialist)	Verified badges, anonymous apply, targeted alerts, calendar scheduler, license verification
Rahul (Nurse)	Mobile quick-apply, shift filters, push notifications, simple profile, local jobs
Suresh (AYUSH)	AYUSH categories, certification upload, availability calendar, locum filters, part-time flags
Priya (HR)	Employer dashboard, candidate pipeline, auto-match, interview scheduler, billing center
Aryan (Recruiter)	Multi-company management, bulk upload, API access, candidate pool search, reporting
Shakir (Admin)	KYC queue, govt CMS, fraud dashboard, audit logs, scheduled publishing
Priyal (Govt Officer)	Admin upload interface, scheduled publish, featured placement, official feed, embargos

---

Job-to-hire funnel & recommended KPIs to track:

1.Discovery: impressions → job views

KPI: job view / impression ratio

2.Interest: job view → save / apply start

KPI: save rate, apply start rate

3.Application: apply start → application submitted

KPI: application completion rate

4.Screening: application submitted → shortlisted

KPI: shortlist rate

5.Interviewing: shortlisted → interview scheduled

KPI: interview scheduling rate, no-show rate

6.Offer: interview → offer

KPI: offer rate

7.Hire: offer accepted

KPI: hire rate, time-to-hire

Instrument each funnel stage and set alerts for drop-offs (e.g., high apply-start but low submission suggests UX friction).

Edge cases & special considerations

Passive candidates: support anonymous profiles that don't appear in public searches unless opted-in.

Cross-country licensing: indicate "license accepted in region X" and allow HR to require specific state/council numbers.

Locum/Shift-based hires: support short-duration contracts, day-by-day availability, and instant-fill bookings.

High-volume govt announcements: use throttled push notifications and staggered job posting strategies to avoid overwhelming servers and users.

Implementation priorities (minimum to delight)

Must-have (MVP): Signup, profile, job search & apply, HR register & KYC queue, admin govt jobs, subscription payments, basic pipeline, email notifications.

Should-have (post-MVP): Resume parsing, calendar sync, anonymous apply, featured listings, basic analytics for HR.

Nice-to-have: ML matching/ranking, background-check integration, enterprise SSO, advanced fraud scoring.

---

4. Panels & features

This section gives an engineering- and product-ready expansion of every panel and subsystem in Medex — user-facing candidate panel, HR/employer panel, and admin panel — plus cross-cutting subsystems (search, payments, messaging, analytics, CMS, security). For each item I list: features, fields/data, UX notes, API hooks, acceptance criteria, priority (MVP / Phase 2 / Later), error/edge cases, and performance/accessibility considerations. Use this as the source for writing stories, API contracts and UI screens.

4.1 Candidate (User) Panel — features & details

Overview

Primary purpose: allow candidates (doctors, nurses, paramedical, AYUSH) to discover jobs, build/verify profiles, apply, and manage job interactions while retaining privacy controls.

Core features (MVP)

Sign up / Sign in

Methods: Email+password, phone OTP, Google/LinkedIn SSO.

Fields: email, phone, name, password, role (candidate default).

UX: progressive sign-up (start with phone/email OTP, complete profile later).

API hooks: `POST /auth/signup`, `POST /auth/otp`.

Acceptance: signup + login successful; email/phone verified.

Edge cases: duplicate emails/phones, OTP resend limits, social auth conflicts.

Profile builder & resume

Sections: Basic info, Contact, Education, Degrees, Specialties, Experience entries, Certifications, Publications, Languages, Notice period, Preferred locations, Salary expectations, Availability/shift preferences, Resume upload.

Resume upload formats: PDF, DOCX, max 10–15 MB.

Resume parsing: auto-populate fields (fallback manual edit).

Privacy toggles: public / masked / private contact visibility.

API: `GET/PUT /profiles/me`, `POST /profiles/me/resume`.

Acceptance: profile saved; resume parsed populates >70% fields for standard CVs.

Edge: parsing failures — allow manual input; virus scan fails — reject.

Medical registration verification

Fields: council name, registration number, certificate image, state/country.

States: unverified / pending / verified / rejected with reason.

UX: show verified badge and date.

API: `/verifications` endpoints; admin webhook for verification results.

Acceptance: admin or automated verification marks status; badge visible.

Edge: fake numbers flagged -> manual review; cross-country license handling.

Job search & filters

Search bar, typeahead suggestions, saved searches, filters: specialty, experience, salary, location & geo-radius, employer type, job type (FT/PT/Contract/Locum), shift (day/night/on-call), posted date, is_featured.

Sort: relevance, newest, salary, distance.

Instant results: use search index for <200ms median.

API: `GET /jobs` with filter params, `GET /search/suggest`.

Acceptance: filter combinations return correct results and pagination.

Job details page

Structured fields: title, employer, verified badge, location, salary range, shift, experience, qualifications, responsibilities, benefits, apply before date, number of openings, contact via platform, job ID & canonical URL, share buttons, similar jobs.

SEO: meta tags, schema.org JobPosting.

Acceptance: pages render SSR with proper metadata.

Apply flow

Options: standard apply (full contact), masked apply (private until shortlisted), quick apply (one-click with default resume), bulk apply (Phase 2).

Attach resume, cover letter, availability notes.

Confirmation UI + email/SMS notification.

API: `POST /jobs/:id/apply`.

Acceptance: application recorded, email sent to user & HR, application visible in candidate's applications list.

Application tracking & history

Track statuses: applied, viewed, shortlisted, interviewed, offered, rejected, withdrawn.

Timeline per application with timestamps & notes.

API: `GET /applications/me`.

Acceptance: status changes notify candidate within X minutes (near real-time).

Notifications & alerts

Channels: email, SMS, push (PWA/web push).

Triggers: new matching job, application status change, interview invitation, admin messages.

Preferences: frequency & channel settings.

Acceptance: notifications delivered reliably; unsubscribe supported.

Saved jobs & alerts

Save job, set job alerts for saved search, daily/real-time digest.

Acceptance: saved jobs persist and alerts fire per preference.

Interview scheduling

Calendar integration (Google / Microsoft) via OAuth, propose/accept slots, generate video link (Zoom/Jitsi) or provide external links.

API: `/interviews` endpoints for scheduling.

Acceptance: calendar invite created correctly; timezone handling accurate.

Candidate Panel — Phase 2 / Later

In-app messaging with attachments and templates.

Resume privacy / anonymized public profile to be discovered by subscribed HRs.

Mentor/CME integrations, salary benchmarking, interview simulator.

Endorsements & references, portfolio links.

Performance & accessibility

Mobile-first UI; offline-resume draft save; WCAG 2.1 AA compliance; screen-reader labels; keyboard accessible forms.

4.2 HR / Employer Panel — features & details

Overview

Employer panel is subscription-gated, requires KYC/admin approval. Focus: create, manage jobs, source candidates, handle pipeline and billing.

Core features (MVP)

Employer registration & KYC

Fields: org name, legal entity type, address, website, GST/PAN/registration docs, authorized signatory, business email (prefer org domain).

KYC docs upload (multi-file), phone verification, admin review notes.

API: `POST /employers/register`, admin endpoints to approve/reject.

Acceptance: admin can process KYC with audit log; rejected reasons stored.

Subscription & billing

Plans: 1m, 3m, 6m, 1y, 2y, 3y, 5y, lifetime (configurable).

Payment methods: card, UPI, netbanking, wallets, enterprise invoices (manual).

Quotas: job slots, featured credits, candidate pool access per plan.

Auto-renew toggles, prorated upgrades/downgrades, coupon support.

API: `/payments/checkout`, `/subscriptions`, webhook handling.

Acceptance: after “payment.succeeded” webhook, subscription activates and quota applied.

Job builder & posting

Structured job builder (mandatory fields): title, specialty tags, location (country/state/city/pincode), salary range, experience, education, license required (yes/no), job type, shift, job description (rich text), benefits, number of openings, apply-by date, screening questions (optional), attachments.

Preview & SEO fields (meta title/description, canonical).

Draft & schedule publish.

Bulk job upload (CSV) — Phase 2.

API: `POST /employers/:id/jobs`.

Acceptance: job published correctly, appears in search, respects visibility & quotas.

Candidate sourcing & pipeline

Candidate list for each job with filters, tags, notes, custom stages (New, Screened, Interview, Offer, Hired).

Auto-match suggestions: top-N candidates with match score and factors (skills, experience, location).

Shortlist & bulk actions (email, schedule interview, reject).

Candidate profile view with resume download & notes.

API: `/employers/:id/candidates`, `/applications/:id/action`.

Acceptance: pipeline actions persist; role-based access enforced.

Interview scheduling & calendar sync

Propose slots, auto-create Google/MS invites, attach video link.

Candidate timezone normalization, reminders & reschedule flow.

Acceptance: invites created and reflect correct time zones.

Team & role management

Employer admin invites teammates with roles: Owner, Recruiter, Viewer, Billing.

Permissions enforced server-side (RBAC).

Acceptance: invited users only access allowed scopes.

Analytics & reporting

Job performance metrics: views, CTR, applications per job, time-to-fill, source of applicants.

Export CSV/PDF reports.

Acceptance: metrics accurate within defined window (e.g., last 30 days).

Support & tickets

Billing support, dispute center, contact support from dashboard.

Acceptance: support ticket created and routed.

HR Panel — Phase 2 / Later

ATS integration, API access for enterprise, SSO (SAML/OAuth), background-check integrations, custom hiring pipelines, branded careers pages.

Performance & UX

Desktop-optimized with data tables and bulk actions. Accessibility for keyboard navigation and screen readers for actions. Fast client-side filtering for pipeline lists.

4.3 Admin Panel — features & details

Overview

Admin controls platform governance: approve HRs, manage govt jobs, moderate content and users, manage plans, view platform metrics and handle disputes.

Core features (MVP)

HR / Employer approval queue

Document viewer, auto-check flags (gmail vs company domain), decision log, reasons codes, SLA timers.

Bulk approve/reject actions.

Acceptance: KYC decisions stored in audit log with actor ID & timestamp.

Govt job CMS

Create/edit/schedule/expire govt jobs; pin to top; mark as authoritative; add structured metadata required for government posts (application portal link, application form, eligibility criteria).

Ability to bulk import Govt notifications (CSV/PDF) and attach scanned documents.

Acceptance: scheduled jobs publish at configured time; featured placement works.

News & content management

Create news, blog posts, site announcements. Tagging, categories, scheduling.

Integrate with homepage feeds and emails.

Acceptance: content publishes with correct metadata and appears in admin feed.

Platform moderation & fraud dashboard

Alerts (suspicious jobs / employers), flag management, manual actions (suspend employer, pause listings, ban user), appeals workflow.

Heuristics and filters: rapid repeat postings, mismatched contact details, high chargeback rate, unusual IP/geolocation patterns.

Acceptance: flagged items appear in queue; admin action reflects across system.

Subscription/plan management

Create/edit plans and coupons, price, features, quotas, associate tax/GST/VAT rates.

Manual invoice generation for enterprise customers.

Acceptance: plan changes reflect on billing pages; new plans available on checkout.

Reporting & analytics

Platform KPIs, revenue reports, churn, fraud incidents, daily active users, job posting volume, search traffic; exportable dashboards.

Acceptance: reports generate within defined SLA.

Audit logs & compliance

Immutable logs of admin actions, KYC decisions, content moderation. Exportable for legal or audit reasons.

Acceptance: logs available and tamper-evident.

Admin — Phase 2 / Later

Role-based admin tiers (content admin, payments admin, security admin), automated KYC integrations, legal hold workflows.

Security & governance

2FA mandatory for admin accounts, IP allow-list for sensitive admin UIs, session audit

4.4 Cross-cutting subsystems (detailed)

A. Search & Indexing

Functionality: full-text search, faceted filters, geo-radius, synonym/abbreviation handling (ENT / Otorhinolaryngology), boost featured/verified employers, fuzzy matching.

Implementation notes: use Elasticsearch or Algolia; event-driven indexing via message queue (job created/updated/deleted → index update).

API: `GET /search`, `POST /search/suggest`, admin `POST /search/reindex`.

Performance: target <200ms median query latency; P95 <500ms.

Acceptance: results relevance tests, synonym mapping tests.

Edge cases: index lag — use eventual-consistency UI messaging for very fresh posts.

B. Payments & Billing

Requirements: sandbox & production with provider(s) (Stripe, Razorpay etc.), webhooks handling, invoice generation (PDF), taxes, refunds, chargeback handling, auto-renewals.

APIs: `/payments/checkout`, `/payments/webhook`, `/subscriptions`, `/invoices`.

Security: PCI-DSS scope reduction via hosted checkout; webhooks signature verification.

Acceptance: successful checkout yields active subscription and posting quota.

Edge cases: webhook retries and idempotency, partial refunds, manual invoice reconciliation.

C. Messaging & Communications

Channels: email (SendGrid/SES), SMS (Twilio/local), push (web push/PWA), in-app messaging.

Features: templated emails (application received, interview invite, payment receipt), transactional & marketing separation, unsubscribes.

API: `/notifications/send`, `/notifications/templates`.

Acceptance: templates customizable; delivery rates measured.

D. Resume parsing & candidate scoring

Functionality: extract degrees, experience, specialties; normalize synonyms; generate candidate match score.

Implementation: off-the-shelf parser (Rchilli/ Sovren) or custom ML pipeline; scoring rules combine skill-match, experience, recency, location.

API: `/parsing/parse`, `/scoring/match-job`.

Acceptance: parsing accuracy benchmarks; match-scoring A/B tests.

E. Analytics & instrumentation

Events: track job.search, job.view, job.apply, application.status_change, payment events, kyc events.

Stack: Segment/PostHog → downstream to GA4, BI (Metabase), data warehouse.

Acceptance: funnels instrumented; dashboards for KPIs.

Privacy: PII redaction in analytics exports.

F. Security & compliance layer

Auth: JWT + refresh tokens; session cookies for web; SSO for enterprise.

RBAC: enforce on API and UI.

Data: encryption at rest, TDE + encrypted S3; retention policies, GDPR-like rights.

Audit: all admin actions logged.

Acceptance: pass OWASP checks, pentest remediation.

G. CDN, caching & performance

Use CDN for static assets; SSR caching for job pages (ISR), edge caching for popular govt pages; Redis cache for session & rate-limiting.

Acceptance: cache hit ratio, TTFB targets.

H. Notifications queue & rate-limits

Use rate-limited notification dispatch to avoid spam; backoff for failed SMS/email; batch digest sending for daily alerts.

4.5 UI component inventory (for frontend devs & designers)

Priority order with short descriptions:

MVP components

Header (auth state, quick nav)

SearchBar with typeahead & location picker

FilterPanel (collapsible)

JobCard (summary)

JobDetail (structured)

ApplyModal (resume select / cover letter / privacy toggle)

ProfileEditor (multi-step wizard)

HR Dashboard (quota, active jobs)

Admin KYC Queue (document viewer)

Notifications Center

Simple Charts (job views / applications / time-to-fill)

Phase 2 components

Candidate Pipeline (Kanban)

Interview Scheduler (slot picker)

In-app Messaging thread

Billing & Invoice viewer (PDF)

Team management (invite modal)

Gov Job List with filters and pinning UI

Micro-interactions

Inline validation, skeleton loaders, optimistic UI for save actions, toast confirmations.

4.6 Acceptance criteria (system-level)

Candidate can discover a job via search and apply with masked option; HR receives application and sees masked identity.

HR signup + KYC + successful payment results in active subscription & job posting quota; admin audit trail records approval.

Admin can post a govt job with scheduled publish that goes live at configured time and triggers notifications.

Search responds <200ms median for common queries under normal load; system handles 3x expected peak for at least 30 minutes (scale plan).

All user-facing forms accessible (WCAG 2.1 AA) and mobile-first

---

4.7 Error handling & resilience (important)

All critical flows (payment, apply) must be idempotent; use idempotency keys on webhooks and checkout.

Provide user-friendly errors: “Payment failed — retry or contact support,” rather than raw server errors.

Queue failed background jobs for retry with exponential backoff and dead-letter queue for manual inspection.

Grace periods: if subscription renewal fails, provide limited grace (e.g., 7 days) before disabling posting.

4.8 Localization & internationalization

Externalize all UI strings (i18n) and support locale-specific formatting (dates, numbers).

Support Right-to-left languages if needed later.

Localize payment methods & tax handling by country.

4.9 Testing notes

Unit tests for business logic (RBAC, quota enforcement, parsing).

Integration tests for API flows (auth, payments, job post, apply).

E2E tests for main user journeys (candidate apply, HR post & shortlist, admin approve).

Accessibility tests (axe) and periodic manual audits.

4.10 Prioritization summary (MVP vs Post-MVP)

MVP must-have: auth, profile/resume upload, basic resume parsing, search & filters, job detail & apply (masked support), HR reg + KYC queue, payments for HR subscriptions (basic), admin govt job post, notifications (email), billing/invoice generation, basic analytics, RBAC, logging.

Phase 2: resume parsing improvement, candidate scoring, in-app messaging, interview scheduler & calendar sync, featured listings, bulk uploads, PWA features and push notifications.

Later: enterprise SSO, ATS integration, advanced ML matching, background-check integration, multi-region legal localization, comprehensive employer branding pages.

4.11 Recommended API surface (quick list)

Auth: `/auth/*`

Profiles: `/profiles/*`

Jobs: `/jobs`, `/jobs/:id`

Applications: `/jobs/:id/apply`, `/applications/:id`

Employers: `/employers/*`

Payments: `/payments/checkout`, `/payments/webhook`

Search: `/search`, `/search/suggest`

Notifications: `/notifications/*`

Admin: `/admin/*` (KYC, govt-jobs, reports)

4.12 KPIs for each panel (what to monitor)

Candidate panel: % profiles completed, apply completion rate, saved search conversion, notification opt-in rates.

HR panel: paid HR conversion rate, jobs posted per HR per month, applications per job, time-to-fill, average spend per HR.

Admin panel: KYC throughput (time to approve), flagged items resolved, fraud incidents per month.

5. End-to-End Flows

Below are the essential end-to-end flows for Medex. Each flow shows actors, step-by-step actions (client & server), API endpoints/DB updates/background jobs involved, success criteria, failure branches and recovery, idempotency/observability notes, and key metrics to instrument. Use these to write stories, design API contracts, and implement automation/monitoring.

Flow A — Candidate signup → profile completion → license verification

Goal: Candidate registers, completes profile, uploads license for verification and begins receiving recommended jobs.

Actors: Candidate (user), Auth service, Profile service, Verification subsystem, Background worker, Admin (if manual verification)

Steps (happy path)

Candidate opens sign-up page and submits sign-up form (email/phone + password or phone OTP).

API: POST /auth/signup or POST /auth/otp

DB: create users row (status: unverified), create audit log.

Event: user.signup

System sends verification OTP/email link.

Background job: send email/SMS.

Event: notification.sent

Candidate verifies email/phone; receives JWT tokens.

API: POST /auth/verify → mark users.verified=true

Event: user.verified

Candidate visits profile builder and uploads resume + fills fields.

API: POST /profiles/me, POST /profiles/me/resume (multipart)

File storage: upload to S3; save profiles.resume_url.

Background job: enqueue resume.parse job with idempotency key.

Resume parsing worker:

Step: virus scan → parse content → normalized fields (degrees, experience, specialties).

DB: update profiles with parsed info; store parsing_result with confidence.

Event: resume.parsed

Candidate enters medical registration number and uploads registration image.

API: POST /verifications (type: medical_license)

DB: create verifications record (status: pending)

Event: verification.requested

Automated verification (if integrated with council API):

Worker calls external council API -> if match → mark verifications.status=verified.

Otherwise: mark pending_manual_review.

Manual admin review (if needed):

Admin UI: view uploaded docs, check council registry

Admin action: PATCH /admin/verifications/:id -> approve/reject.

DB: update verifications status; update profiles.verified=true if approved.

Notifications: candidate receives email about verification result.

Success criteria

Candidate can complete profile and see parsed fields.

License verification moves to verified (or pending_manual_review) with notification.

Candidate appears in candidate pool search if privacy permits.

Failure branches & recovery

Upload fails (network/virus): return user-friendly error; retry allowed. Resume parse failing → mark parsing_result.errors, show manual edit UI.

External council API unavailable: mark verification.status=queued_retry and retry with exponential backoff; notify candidate of delay.

Fake/invalid license: admin rejects; candidate receives reason and steps to appeal.

Idempotency & security

Use idempotency keys on resume upload and verification submission.

Store uploads in S3 with server-side encryption; access controlled.

Log every admin action (audit log).

Observability & metrics

Events: user.signup, profile.complete, resume.parsed, verification.status_change.

KPIs: profile completion rate, license verification success rate, parsing success rate, time-to-verify.

Flow B — HR (Employer) registration → KYC → admin approval → payment → subscription activation

Goal: Employer signs up, completes KYC, gets approved by admin, pays for a subscription, and gains posting quota.

Actors: HR user, Employer service, Admin, Payments service, Background worker

Steps (happy path)

HR submits employer registration form with KYC docs.

API: POST /employers/register (fields + docs)

DB: create employers row with status pending_kyc; store docs in secure S3.

Event: employer.registration_submitted

System triggers automated checks: domain email vs website, phone OTP, WHOIS checks.

Background worker runs heuristics and sets employers.risk_score.

Event: employer.kyc_auto_check_completed

Admin reviews KYC in Admin Panel; approves or rejects.

API: PATCH /admin/employers/:id/approve or reject

DB: update employers.status = approved or rejected

Notifications: HR gets email about status.

Upon approval HR lands on Billing page; chooses plan (e.g., 3 months).

UI: shows plan details, quotas and tax calculation.

HR starts checkout -> payment provider (hosted checkout or tokenized card) created.

API: POST /payments/checkout → returns checkout_session_id or redirect URL.

Client: redirect to hosted checkout (or inline form).

Payment provider completes payment and sends webhook payment.succeeded.

API: POST /payments/webhook (verify signature).

Worker/process: Validate webhook idempotently; create payments record and subscriptions record: set subscription.status = active, set employers.subscription_id.

Apply quotas: increment employers.quota.job_slots = plan.job_slots.

Notification: send invoice (email) and dashboard update.

HR can now create jobs up to quota.

Success criteria

Employer status becomes approved.

Payment succeeded processed and subscription active with quotas applied.

Admin and HR receive receipts and audit entries.

Failure branches & recovery

KYC failed: admin rejects -> HR can resubmit docs. Track attempts to prevent abuse.

Payment failed: show detailed error; schedule retry or show retry link. Use webhook to handle asynchronous success.

Duplicate webhook events: verify webhook event_id and store processed event IDs to ensure idempotency.

Chargeback/refund: flag subscription, notify admin, possibly suspend posting until resolved.

Idempotency & security

Webhook handling must be idempotent (store gateway_event_id unique).

Sensitive KYC docs encrypted; access controlled; purge per retention policy if rejected.

PCI scope minimized by using hosted checkout or tokenization.

Observability & metrics

Events: employer.kyc_submitted, employer.kyc_approved, payment.success, subscription.activated

KPIs: time-to-approve KYC, conversion rate HR signup → paid, payment success rate, churn.

Flow C — HR creates job (draft → publish) → indexing → candidate notifications

Goal: HR posts a job; job is validated, stored, indexed, and matching candidates are notified.

Actors: HR, Jobs service, Search indexer, Notification service, Background workers

Steps (happy path)

HR builds job in Job Builder and saves as draft.

API: POST /employers/:id/jobs (status: draft)

DB: create jobs row with status=draft. Save structured fields and attachments.

Event: job.created

HR reviews and clicks Publish.

API: PATCH /jobs/:id with status=published (server validates quota and KYC/paid status).

DB: update jobs.status=published, set posted_at.

If quota exceeded: return 422.

On publish, a message is pushed to indexing queue -> job_indexer worker.

Worker: generate index document, apply synonyms/normalization, push to Elasticsearch/Algolia.

Event: job.indexed

After indexing success, notification worker runs matching against candidate profiles (auto-match) and:

Add to candidate digest queue or push real-time notifications depending on user preference.

Event: job.match_candidates_enqueued

Candidate notifications sent (email/SMS/push/digest).

Throttling: batch per candidate, group similar jobs to avoid spam.

Event: notification.sent per channel.

Success criteria

Job visible in search and job detail page.

Indexing completes and job appears within search latency target.

Matching candidates receive notifications according to preferences.

Failure branches & recovery

Indexing failure: job stays published but flagged indexed=false and queued for retry; admin alerted if persistent failure.

Notification failures (email bounce): mark candidate contact as invalid and flag for cleanup; fallback to in-app notification.

Quota enforcement: if publish attempted beyond quota, return clear error and link to upgrade.

Idempotency & security

Indexer must handle duplicate indexing requests idempotently (use job ID).

Sanitise job content to avoid XSS in rendered HTML.

Observability & metrics

Events: job.published, job.indexed, notifications.sent.

KPIs: index latency, notification delivery rate, applications generated per job.

Flow D — Candidate view job → apply (masked vs open) → application tracking

Goal: Candidate views job details and applies. Application is delivered to HR, status tracked.

Actors: Candidate, Jobs service, Applications service, HR, Notifications service

Steps (happy path: masked apply)

Candidate views job details (SSR) -> click Apply (Masked).

Client: gather resume_id, cover letter, privacy setting apply_type=masked.

API: POST /jobs/:id/apply with apply_type=masked, resume_id.

DB: create applications row (status=applied, visibility=masked), store application_id.

Event: application.created

Application triggers:

Notification to candidate: application.received.

Notification to HR: application.new (HR sees masked profile; contact fields anonymized).

HR reviews in pipeline and decides to request_contact.

API: POST /applications/:id/request_contact (HR action)

System creates contact_release_request for candidate.

Candidate is notified of request and can choose to release contact.

If candidate approves, applications.visibility=unmasked and HR receives real contact info.

Status transitions (HR shortlists -> interview -> offer etc.) update applications with events and notify both parties.

Steps (open apply)

If apply_type=open, HR receives full contact details immediately and pipeline proceeds faster.

Success criteria

Application is created and visible in both candidate and HR dashboards.

Masked apply respects privacy settings and contact flow works.

Failure branches & recovery

Resume attachment failure: rollback application and notify candidate to retry.

Duplicate application: reject with message (use unique constraint: user_id + job_id) or allow re-apply with different resume and mark as reapplied.

HR not responding: candidate can withdraw application; candidate gets an auto-reminder after N days.

Idempotency & security

Use idempotency key for apply request to avoid duplicate applications.

Sanitize cover letters and attachments.

Observability & metrics

Events: application.created, application.status_change

KPIs: application completion rate, masked-release conversion, average HR response time.

Flow E — Resume parsing pipeline (upload → parse → result)

Goal: Parse uploaded resume into structured candidate profile fields to enable search & matches.

Actors: Candidate, Storage (S3), Virus scan service, Parser (Rchilli/Sovren/custom), Worker, Profiles DB

Steps (happy path)

Candidate uploads resume via POST /profiles/me/resume.

API stores file in S3 and enqueues resume.parse job (payload: resume_url, user_id, request_id).

Event: resume.uploaded

Worker pulls job:

Step 1: virus scan (ClamAV/managed) — if fail, mark parse_status=virus_detected.

Step 2: call parser API or run local ML parser.

Step 3: normalize fields (degrees, positions, dates, companies, skills), map synonyms.

Step 4: save parsed JSON in DB profiles.parsed_resume.

Post-parse:

If confidence high: update profile fields automatically and notify candidate to review.

If low-confidence: mark parsing_status=requires_review and show candidate the extracted fields for manual confirmation.

Failure branches & recovery

Parser timeout or 5xx: retry with exponential backoff, dead-letter after N attempts; create manual review ticket.

Corrupted file: notify candidate and ask to upload again.

Idempotency & security

Use job idempotency; only perform parse once per resume_version_id.

Delete resume from worker if fails continuously; mark for admin review.

Observability & metrics

Events: resume.parse.started, resume.parse.completed, resume.parse.failed.

KPIs: parse success rate, average parse time, % auto-filled fields.

Flow F — Interview scheduling (HR proposes → candidate accepts → calendar sync → reminders)

Goal: Schedule an interview between HR and candidate with calendar invites and reminders.

Actors: HR, Candidate, Scheduler service, Calendar providers (Google/Microsoft), Notification service

Steps (happy path)

HR selects candidate and clicks Schedule interview -> proposes time slots.

API: POST /interviews/propose with one or more ISO8601 slots.

DB: create interview record with status=proposed, store proposed_slots.

Event: interview.proposed

Candidate receives notification/push and clicks Accept slot.

API: POST /interviews/:id/accept with chosen slot.

DB: update interview.status = scheduled, scheduled_at = chosen_slot.

System optionally creates calendar invites:

If HR and/or candidate connected calendar via OAuth:

Scheduler calls Google/Microsoft Calendar API to create event and guest invites.

If successful, store calendar_event_id.

Notifications: emails with ICS attachments and video link (if provided).

Reminders: schedule reminder notifications (24h, 1h).

Worker enqueues reminder jobs.

After interview completed, HR updates status (no-show / completed / needs follow-up).

Failure branches & recovery

Calendar OAuth expired: prompt user to re-authenticate and fallback to sending ICS as attachment.

Timezone mismatch: system normalizes to user timezone and shows local time in UI.

Candidate declines all slots: HR can propose new slots or request reschedule.

Idempotency & security

Use idempotent creation for calendar events (store idempotency token).

Ensure calendar events are private and only accessible to participants.

Observability & metrics

Events: interview.proposed, interview.scheduled, interview.reminder.sent, interview.no_show.

KPIs: interview scheduling time, no-show rate, calendar sync success rate.

Flow G — Offer issuance → acceptance (e-sign optional) → job closure

Goal: HR issues offer; candidate accepts (optionally e-signs); job is marked filled and candidate/job statuses updated.

Actors: HR, Candidate, Files service, Notifications, Admin (optional)

Steps (happy path)

HR uploads offer letter PDF and marks candidate as offered with compensation, start date, and conditions.

API: POST /applications/:id/offer (attach offer document)

DB: update applications.status = offered, store offer sub-object.

Notification: candidate notified.

Candidate views offer and clicks Accept (or Negotiate/Decline).

API: POST /applications/:id/offer/accept or /decline.

If e-sign enabled: candidate e-signs using a signature widget (or third-party e-sign provider) and signed document is stored with audit trail.

DB: applications.status = accepted, hired_at = now()

On acceptance:

HR marks job as filled and decrements employer.quota accordingly.

System may prompt candidate to mark as hired and ask for feedback/review.

Notifications: both parties receive final confirmation and invoice (if applicable).

Optionally, platform records a placement (for metrics/revenue) and may trigger placement fee if model includes success fee.

Failure branches & recovery

Candidate disputes offer: create support ticket; admin mediation.

Offer expired: move to offer.expired and notify HR.

Observability & metrics

Events: offer.created, offer.accepted, job.filled

KPIs: offer acceptance rate, time-from-offer-to-accept, hires per employer.

Flow H — Admin posts government job (schedule → pre-warm → publish → handle traffic)

Goal: Admin publishes authoritative govt job; platform handles scheduled release & large traffic spikes.

Actors: Admin, CMS, CDN, Notification service, Cache pre-warm workers

Steps (happy path)

Admin creates government job in CMS and schedules publish time (e.g., 10:00 AM on X date).

API: POST /admin/govt-jobs with publish_at.

DB: create govt_jobs record (status: scheduled).

Pre-warm tasks scheduled ahead of publish (e.g., 30 min prior):

Worker pre-renders SSR page and uploads to CDN origin or creates cache entries.

Prepare notification batch and compute target candidate segments.

Event: govt_job.prewarm_complete

On publish time:

CMS atomically flips status=published, clears any cache invalidations and triggers cache warming to CDN edge.

Notifications: fire urgent notifications to subscribed candidate segments using throttled bursts (divide into batches to respect rate limits).

Event: govt_job.published

Monitor traffic and scale:

Autoscale web nodes & search cluster; monitor response times.

Queue any heavy operations (parsing, admin reports) to background workers.

Failure branches & recovery

CDN propagation delay: show fallback page with scheduled publish time; retry warming tasks.

Notification spike causing throttling: use exponential backoff + status page notice; provide digest fallback.

Spam or fake govt job detected: admin can unpublish and post correction notice; track impacted candidates.

Idempotency & security

Pre-warm tasks use job IDs for idempotency; notification batches store sent IDs to avoid duplicates.

Govt job fields require admin 2FA to prevent accidental publishes.

Observability & metrics

Events: govt_job.scheduled, govt_job.prewarm, govt_job.published

KPIs: peak concurrency, cache hit ratio for govt pages, notification delivery timeliness.

Flow I — Payment lifecycle: checkout → webhook → renewals → failed payments → refunds

Goal: Robust payment handling for subscriptions, including webhooks, retries, proration, cancellations and refunds.

Actors: HR, Payments gateway, Billing service, Background worker, Admin

Steps (happy path)

HR chooses plan and initiates checkout.

API: POST /payments/checkout -> returns checkout_session (hosted or tokenized).

Client: completes checkout.

Payment gateway sends payment.success webhook.

API: POST /payments/webhook with signature header.

Server: verify signature, look up checkout_session_id, create payments record (status succeeded), create subscriptions record with status=active, set starts_at, expires_at.

Apply quotas and send invoice email.

Event: subscription.activated

Auto-renew:

Background worker scheduled n days before expiry attempts renewal via saved payment method (if auto-renew enabled).

On success: update subscriptions.expires_at, emit subscription.renewed.

On failure: set subscription.renewal_attempts++, send failure email and retry schedule; after max_retries suspend posting privileges and notify admin.

Cancellation:

HR may cancel subscription via UI -> POST /subscriptions/:id/cancel. Policy: active until expires_at or immediate depending on terms.

Refunds: POST /payments/:id/refund (admin/manual or gateway).

Chargebacks:

On payment.chargeback webhook: mark payments.status = chargeback, notify finance team, freeze employer pending investigation.

Failure branches & recovery

Webhook replay/duplicate: store gateway_event_id and skip if seen.

Payment provider outage: queue webhook events and reconcile later; show UI warning; allow manual invoice payment option.

Disputed refunds: route to admin and record decision/communication.

Idempotency & security

Webhook idempotency via gateway_event_id.

Do not store raw card data; store gateway token only.

Use secure server-to-server verification and signed webhooks.

Observability & metrics

Events: payment.created, payment.succeeded, payment.failed, subscription.renewed, refund.processed

KPIs: payment success rate, renewal rate, churn, D0 payment failures.

Flow J — Fraud detection → admin review → suspension → appeal

Goal: Detect suspicious employers/jobs/candidates and handle via admin moderation and user communication.

Actors: Fraud detection engine (heuristics/ML), Admin, Employer/Candidate, Notifications

Steps (happy path detection & resolution)

Heuristics flag entity (e.g., employer uses gmail, many postings same day, phone mismatch).

Event: fraud.flagged with entity_id, reason_codes.

DB: create fraud_reports record.

Automatic protective actions (depending on risk score):

Soft block: pause posting for the employer and set status=suspended_temp.

Create support ticket and notify the admin team with priority.

Notify employer with high-level reason (not disclosing heuristics).

Admin reviews flagged account in Admin Panel:

Admin can unflag (restore), suspend (long-term), or ban.

Record decision with audit_log and notify employer/candidate of outcome.

Appeals:

Employer submits appeal with supporting docs.

Admin reevaluates and decides; if reinstated, publish notice.

Failure branches & recovery

False positives: allow rapid appeal and manual override; track false-positive rate and tune heuristics.

Malicious actors evading detection: add rate-limits, 2FA, and manual escalation.

Idempotency & security

All suspension actions logged and require admin 2FA for escalations.

Limit auto-suspension to high confidence cases; otherwise, set temp_hold and require manual review.

Observability & metrics

Events: fraud.flagged, fraud.actioned

KPIs: fraud detection rate, false positive rate, avg time to resolution.

Flow K — Full search reindex (admin-triggered)

Goal: Rebuild search index for jobs/candidates (full reindex) with minimal user disruption.

Actors: Admin, Indexer service, Message queue, Search cluster (ES/Algolia), Monitoring

Steps

Admin clicks Reindex in Admin Panel -> POST /admin/search/reindex

Server: create reindex_job entry (status: queued), snapshot current index metadata.

Reindex job worker:

Splits work into batches (by created_at ranges or IDs) and enqueues index.batch jobs.

Each batch worker fetches raw documents from DB, transforms and writes to search index.

Use zero-downtime pattern: write to a new index alias and swap alias when complete.

Health checks & validation:

After indexing, run sample queries comparing old vs new results to ensure parity.

Swap aliases atomically and mark reindex_job.status = completed.

Rollback:

If validation fails, do not swap alias; mark reindex_job.status = failed, send admin alert with logs.

Failure branches & recovery

Partial failures: retries for failing batches; skip problematic docs to dead-letter queue for manual investigation.

Long-running job: queue backpressure, autoscale workers.

Observability & metrics

Events: reindex.started, reindex.batch.completed, reindex.completed

KPIs: time for reindex, batch failure rate, index size.

Flow L — HR bulk job upload (CSV) → validate → create jobs

Goal: HR/Agency uploads many jobs in a CSV; system validates rows, creates draft jobs, and reports errors.

Actors: HR, Upload service, CSV validator worker, Jobs service

Steps

HR uploads CSV file via POST /employers/:id/jobs/bulk-upload.

API: store file in S3, create bulk_upload record with status = uploaded.

Event: bulk_upload.received

Validator worker runs:

Parse CSV, validate required columns, validate fields (specialty exists, salary numeric), detect duplicates.

Produce validation_report (line errors/warnings).

For valid rows, create jobs records in draft state using transactions; for invalid rows, include errors in report.

On completion, notify HR with validation_report and link to import preview.

HR reviews and confirms to publish (or schedule).

Publish step follows regular job.publish flow (quota check + indexing).

Failure branches & recovery

Malformed CSV: return parse error with guidance.

Partial import: only valid rows created; invalid rows reported.

Observability & metrics

Events: bulk_upload.completed, bulk_upload.errors

KPIs: bulk upload success rate, avg rows per CSV, error types.

Cross-cutting Observability & Operational Notes

Correlation IDs: Generate a request/trace id (X-Request-ID) and propagate across services, queues and workers to ease debugging.

Idempotency tokens: Use idempotency tokens for payment checkouts, apply actions, and webhook handling.

Audit logs: All admin and critical actions must be recorded immutably with actor, timestamp and context.

Message bus: Use durable queue (e.g., RabbitMQ, SQS, or Kafka) with dead-letter queues and retry policies.

Monitoring & alerting: Track SLOs (search latency, payment success rate), set alerts for anomalies (e.g., sudden drop in payment webhooks, spike in fraud flags).

Backups & DR: Regular DB snapshots, test restores; cross-region replication for critical services.

Security: Restrict sensitive admin endpoints behind IP allowlists + 2FA; rotate secrets.

Acceptance Test Checklist (applies across flows)

Sign-up & login flows work for all roles and social logins.

Resume uploads accepted and parsed as per SLA (e.g., parse within 60s for typical CV).

KYC approval path allows admin to approve/reject and the decision propagates to employer view.

Payment checkout results in subscription.active and quota applied via webhook idempotently.

Publishing a job causes index update and the job becomes searchable within defined latency.

Apply flow creates application and notifies HR; masked apply hides contact info until release.

Interview scheduling creates calendar events and reminders in correct timezones.

Govt job scheduled publish works and pre-warm reduces TTFB at go-live.

---

6 . Data model & schema notes (DDL-style examples)

You already have a solid core schema. Below I expand it with practical, production-ready SQL artifacts, partitioning examples, triggers, maintenance queries, reporting views, GDPR/anonymization helper, and recommended privileges. Use these directly in your DB repo as migration templates (Postgres 13+ with PostGIS).

6.1 Additional tables & utilities (DDL)

job_versions — track edits to jobs (audit/history)

```
CREATE TABLE job_versions (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  job_id UUID NOT NULL REFERENCES jobs(id) ON DELETE CASCADE,
  changed_by UUID REFERENCES users(id),
  change_type TEXT NOT NULL, -- 'create','update','publish','close'
  payload JSONB NOT NULL,    -- full snapshot or diff
  created_at TIMESTAMPTZ DEFAULT now()
);

CREATE INDEX idx_job_versions_job ON job_versions(job_id, created_at DESC);
```

application_audit — immutable application status changes & notes

```
CREATE TABLE application_audit (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  application_id UUID REFERENCES applications(id) ON DELETE CASCADE,
  actor_id UUID, -- who made the change (hr / admin / system)
  from_status TEXT,
  to_status TEXT,
  note TEXT,
  metadata JSONB,
  created_at TIMESTAMPTZ DEFAULT now()
);

CREATE INDEX idx_application_audit_app ON application_audit(application_id);
```

candidate_scores — cached match scores per job-candidate (denormalized for fast HR match)

```
CREATE TABLE candidate_scores (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  job_id UUID REFERENCES jobs(id) ON DELETE CASCADE,
  candidate_profile_id UUID REFERENCES profiles(id) ON DELETE CASCADE,
  score NUMERIC(5,2) NOT NULL, -- 0.00 - 100.00
  reasons JSONB,               -- e.g.
{"skill_match":40,"experience":30,"location":30}
  computed_at TIMESTAMPTZ DEFAULT now()
);

CREATE INDEX idx_candidate_scores_job ON candidate_scores(job_id);
CREATE INDEX idx_candidate_scores_candidate ON
candidate_scores(candidate_profile_id);
```

6.2 Example triggers / functions (automatic counters & de-normalization)

1) Trigger to increment jobs.view_count safely (uses lightweight UPDATE)

```
CREATE OR REPLACE FUNCTION job_view_counter()
RETURNS TRIGGER AS $$
BEGIN
  UPDATE jobs
  SET view_count = view_count + 1, updated_at = now()
  WHERE id = NEW.job_id;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TABLE job_views (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  job_id UUID REFERENCES jobs(id),
  session_id TEXT,
  user_id UUID,
  created_at TIMESTAMPTZ DEFAULT now()
);

CREATE TRIGGER trg_job_view
AFTER INSERT ON job_views
FOR EACH ROW EXECUTE PROCEDURE job_view_counter();
```

> Note: For very high traffic, write view events to a queue or Redis counter and flush periodically to DB to avoid hot-row contention.

2) Trigger to maintain jobs.apply_count

```
CREATE OR REPLACE FUNCTION update_job_apply_count()
RETURNS TRIGGER AS $$
BEGIN
  IF (TG_OP = 'INSERT') THEN
    UPDATE jobs SET apply_count = apply_count + 1 WHERE id = NEW.job_id;
    RETURN NEW;
  ELSIF (TG_OP = 'DELETE') THEN
    UPDATE jobs SET apply_count = GREATEST(coalesce(apply_count,0) - 1, 0) WHERE
id = OLD.job_id;
    RETURN OLD;
  END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_applications_insert
AFTER INSERT ON applications
FOR EACH ROW EXECUTE PROCEDURE update_job_apply_count();

CREATE TRIGGER trg_applications_delete
AFTER DELETE ON applications
FOR EACH ROW EXECUTE PROCEDURE update_job_apply_count();
```

6.3 Partitioning examples (monthly partition for jobs by posted_at)

```
-- Parent table exists already (jobs). Convert to partitioned table (requires
special care if populated).
-- Example to create partitions for 2025 Jan - Mar
CREATE TABLE jobs_2025_01 PARTITION OF jobs
  FOR VALUES FROM ('2025-01-01') TO ('2025-02-01');

CREATE TABLE jobs_2025_02 PARTITION OF jobs
  FOR VALUES FROM ('2025-02-01') TO ('2025-03-01');

CREATE TABLE jobs_2025_03 PARTITION OF jobs
  FOR VALUES FROM ('2025-03-01') TO ('2025-04-01');

-- Use automation (cron job or migration) to create future partitions.
```

Recommendation: Partition by month for `jobs` and `applications` in high-volume deployments. Keep an automated partition manager script in infra.

6.4 Materialized views & reporting (example view: daily job applications)

```
CREATE MATERIALIZED VIEW mv_daily_applications AS
SELECT
  date_trunc('day', created_at) AS day,
  job_id,
  count(*) AS applications
FROM applications
GROUP BY date_trunc('day', created_at), job_id
WITH NO DATA;

-- Refresh nightly or via job:
REFRESH MATERIALIZED VIEW CONCURRENTLY mv_daily_applications;
```

6.5 GDPR / Right-to-be-forgotten helper function (anonymize profile)

Caution: This should be run only after legal confirmation; store deletion requests/audit.

```
CREATE OR REPLACE FUNCTION anonymize_profile(p_user_id UUID)
RETURNS VOID AS $$
BEGIN
  -- Soft-delete personal PII
  UPDATE users SET
    email = concat('deleted+', gen_random_uuid()::text, '@redacted.local'),
    phone = NULL,
    is_email_verified = FALSE,
    is_phone_verified = FALSE,
    updated_at = now()
  WHERE id = p_user_id;

  UPDATE profiles SET
    full_name = NULL,
    display_name = NULL,
    dob = NULL,
    headline = NULL,
    about = NULL,
```

```
    registration_numbers = '[]'::jsonb,
    resume_url = NULL,
    resume_checksum = NULL,
    profile_visibility = 'private',
    is_deleted = TRUE,
    updated_at = now()
  WHERE user_id = p_user_id;

  -- Optionally write a deletion record
  INSERT INTO audit_logs(actor_id, actor_role, action_type, entity_type,
entity_id, details)
    VALUES (NULL, 'system', 'gdpr_anonymize', 'user', p_user_id,
jsonb_build_object('note','anonymized per user request'));

END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
```

6.6 Backups & restore (psql examples)

Logical backup (daily):

```
PGPASSWORD=$PG_PASS pg_dump -h $DB_HOST -U $DB_USER -Fc medex_prod >
/backups/medex_prod_$(date +%F).dump
```

Point-in-time recovery: enable WAL archiving and use base backups + WAL segments for RPO.

Test restore script (nightly in CI to validate backups restore successfully).

6.7 Grants & roles (least privilege)

```
-- Create roles
CREATE ROLE medex_app;
CREATE ROLE medex_readonly;

-- Grant usage to app role (assuming app connects with medex_app)
GRANT CONNECT ON DATABASE medex_prod TO medex_app;
GRANT USAGE ON SCHEMA public TO medex_app;

-- Grant specific privileges to the app role (example)
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE users, profiles, jobs, applications,
employers TO medex_app;
GRANT SELECT ON ALL SEQUENCES IN SCHEMA public TO medex_app;

-- Readonly for analytics
GRANT CONNECT ON DATABASE medex_prod TO medex_readonly;
GRANT USAGE ON SCHEMA public TO medex_readonly;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO medex_readonly;
```

Rotate credentials using your secrets manager and don't embed plaintext DB creds in code.

6.8 Example useful queries (production ops)

Find long-running transactions:

```
SELECT pid, usename, query, state, now() - query_start AS duration
FROM pg_stat_activity
WHERE state <> 'idle'
ORDER BY duration DESC;
```

Check bloat on table

```
SELECT schemaname, tablename, bs*tblpages AS table_size_bytes, tblpages, est_rows
FROM (
  SELECT schemaname, tablename, tblpages,
         pg_relation_size(quote_ident(schemaname) || '.' ||
quote_ident(tablename)) / current_setting('block_size')::int AS bs
  FROM pgstattuple('public.jobs')
) t;
```

(Use pg_repack or VACUUM FULL with maintenance windows)

6.9 Index strategy checklist

GIN `to_tsvector` for full-text `jobs(title || description)`.

GIN on `specialty` (text array).

GIST index for PostGIS `location_geog` for radius queries.

B-tree indexes for common equality/filter columns: `employer_id`, `status`, `posted_at`.

Partial indexes for hot queries: e.g., published jobs only:

```
CREATE INDEX idx_jobs_published_postedat ON jobs (posted_at) WHERE status =
'published';
```

6.10 Migration best-practices (practical steps)

Add new nullable columns first; deploy code that writes to both old & new columns (dual-write if needed).

Backfill in batches via a worker (e.g., 10k rows per job).

Switch reads to new columns in a feature-flagged deploy.

Remove old columns in a subsequent migration after verification.

Avoid `ALTER TABLE ... DROP COLUMN` during peak without maintenance window.

6.11 Monitoring & maintenance queries to schedule

Monitor table growth:

```
SELECT schemaname, relname, pg_size_pretty(pg_total_relation_size(oid)) FROM
pg_class c JOIN pg_namespace n ON n.oid = c.relnamespace WHERE relkind = 'r'
ORDER BY pg_total_relation_size(oid) DESC LIMIT 20;
```

Monitor index usage:

```
SELECT relname, idx_scan, idx_tup_read, idx_tup_fetch FROM pg_stat_user_indexes
JOIN pg_index ON pg_stat_user_indexes.indexrelid = pg_index.indexrelid ORDER BY
idx_scan DESC LIMIT 50;
```

Schedule: nightly VACUUM (autovacuum tuned), weekly reindex on low-traffic windows for fragmented indexes, monthly analyze.

6.12 ER diagram (text Mermaid snippet you can paste into mermaid.live)

```
erDiagram
  USERS ||--o{ PROFILES : has
  USERS ||--o{ EMPLOYER_USERS : belongs
  EMPLOYERS ||--o{ EMPLOYER_USERS : employs
  EMPLOYERS ||--o{ JOBS : posts
  JOBS ||--o{ APPLICATIONS : receives
  APPLICATIONS ||--o{ APPLICATION_AUDIT : audited_by
  JOBS ||--o{ JOB_VERSIONS : versioned
  JOBS ||--o{ CANDIDATE_SCORES : scored
  PROFILES ||--o{ CANDIDATE_SCORES : scored_for
  APPLICATIONS ||--o{ INTERVIEWS : schedules
  USERS ||--o{ NOTIFICATIONS : receives
  USERS ||--o{ AUDIT_LOGS : performs
```

6.13 Example migration skeleton (SQL file names + ordering)

`001_create_users_profiles_employers.sql` — core tables

`002_create_jobs_applications.sql` — jobs & applications

`003_create_payments_subscriptions.sql` — payments & plans

`004_add_search_indexes.sql` — GIN/GIST indexes

`005_create_job_versions_and_audit.sql` — audit tables

`006_partition_jobs.sql` — create partitioning function & initial partitions

`007_triggers_counters.sql` — triggers (view/apply counts)

`008_materialized_views.sql` — reporting MVs

Run migrations in CI with a dry-run stage and have a rollback plan.

6.14 Short checklist before production DB go-live

✅ Encrypt connections (force SSL) and disable password authentication from public IPs.

✅ Grant minimal DB privileges to app role.

✅ Configure automated backups & WAL archiving.

✅ Set autovacuum parameters tuned for your workload (large inserts vs OLTP read-heavy).

✅ Provision read replicas for analytics and heavy read traffic.

✅ Set up monitoring dashboards for DB metrics (CPU, I/O, connections, replication lag).

✅ Test restore from backup quarterly.

7. API DESIGN

7.1 API Fundamentals

Base URL:
`https://api.medex.health/v1/`

Auth:
Bearer token via JWT (`Authorization: Bearer <token>`)

Content type:
`application/json` for request/response

Versioning:
URI versioning (`/v1/`), optional header `X-API-Version`

Pagination:
`?page=1&limit=20` with `X-Total-Count` header

Error format:

```
{
        "error": {
          "code": "RESOURCE_NOT_FOUND",
          "message": "Job not found",
          "details": { "job_id": "12345" }
        }
      }
```

7.2 AUTH & USERS

POST /auth/register

Registers a new user.

Request

```
{
  "email": "dr.sara@example.com",
  "password": "StrongPass!1",
  "role": "candidate",
  "full_name": "Dr. Sara Mehta"
}
```

Response

```
{
  "user": {
    "id": "b73a1c92-e6b2-41af-b1e1-04c93a87dfc8",
    "email": "dr.sara@example.com",
    "role": "candidate"
  },
```

```
  "token": "eyJhbGciOiJIUzI1..."
}
```

Errors

```
EMAIL_ALREADY_EXISTS

INVALID_PASSWORD_POLICY
```

POST /auth/login

Authenticates a user and returns JWT.

Request

```
{
  "email": "hr@apollohospital.com",
  "password": "securePass"
}
```

Response

```
{
  "token": "eyJhbGciOiJIUzI1...",
  "user": {
    "id": "f3aa9bcd-32e1-4d02-b2e5-f44bce11b1ff",
    "role": "employer"
  }
}
```

GET /users/me

Returns authenticated user details.

Response

```
{
  "id": "b73a1c92-e6b2-41af-b1e1-04c93a87dfc8",
  "email": "dr.sara@example.com",
  "role": "candidate",
  "created_at": "2025-01-14T08:22:00Z"
}
```

7.3 CANDIDATE PROFILE

GET /profiles/me

Get own profile.

Response

```
{
  "id": "a79a1b31-3af3-46a2-b17d-77e9a32f36b3",
```

```
  "user_id": "b73a1c92-e6b2-41af-b1e1-04c93a87dfc8",
  "full_name": "Dr. Sara Mehta",
  "specialization": ["Pediatrics", "Neonatology"],
  "experience_years": 5,
  "preferred_location": ["Jaipur", "Delhi"],
  "resume_url": "https://cdn.medex.health/resumes/sara.pdf"
}
```

PATCH /profiles/me

Update profile.

Request

```
{
  "specialization": ["Pediatrics", "Child Health"],
  "preferred_location": ["Delhi"]
}
```

7.4 JOB MANAGEMENT (Employer)

POST /employers/{employer_id}/jobs

Create a new job post.

Request

```
{
  "title": "Resident Doctor - Pediatrics",
  "description": "Responsible for NICU ward duties...",
  "location": "Jaipur",
  "specialization": ["Pediatrics"],
  "min_experience": 2,
  "salary_range": "₹60,000–₹90,000",
  "status": "draft"
}
```

Response

```
{
  "id": "job_7d21e1c4",
  "status": "draft",
  "posted_at": null
}
```

Errors

```
EMPLOYER_NOT_VERIFIED

INVALID_FIELD
```

PATCH /jobs/{id}

Edit an existing job.

Request

```
{
  "status": "published"
}
```

Response

```
{
  "id": "job_7d21e1c4",
  "status": "published",
  "posted_at": "2025-10-12T11:00:00Z"
}
```

GET /jobs?status=published&specialization=Pediatrics

Public job listings (search & filter).

Response

```
{
  "page": 1,
  "limit": 2,
  "total": 52,
  "data": [
    {
      "id": "job_7d21e1c4",
      "title": "Resident Doctor - Pediatrics",
      "hospital": "Apollo Jaipur",
      "location": "Jaipur",
      "salary_range": "₹60,000–₹90,000",
      "posted_at": "2025-10-10T09:00:00Z"
    },
    {
      "id": "job_7d21e1c5",
      "title": "Consultant Pediatrician",
      "hospital": "Fortis Delhi",
      "location": "Delhi"
    }
  ]
}
```

7.5 JOB APPLICATION FLOW

POST /jobs/{job_id}/apply

Apply to a job.

Request

```
{
  "profile_id": "a79a1b31-3af3-46a2-b17d-77e9a32f36b3",
  "cover_letter": "I have 5 years NICU experience...",
  "resume_url": "https://cdn.medex.health/resumes/sara.pdf"
}
```

Response

```
{
  "application_id": "app_9c43e1f9",
  "status": "submitted",
  "created_at": "2025-10-12T11:21:00Z"
}
```

GET /applications/{id}

Get application details.

Response

```
{
  "id": "app_9c43e1f9",
  "job_id": "job_7d21e1c4",
  "status": "under_review",
  "feedback": "Shortlisted for interview",
  "interview_date": "2025-10-18T10:00:00Z"
}
```

7.6 EMPLOYER DASHBOARD

GET /employers/{id}/dashboard

Analytics for an employer’s jobs.

Response

```
{
  "total_jobs": 25,
  "total_applications": 740,
  "active_jobs": 8,
  "avg_time_to_hire_days": 18,
  "top_specialties": ["Anesthesiology", "Pediatrics"]
}
```

7.7 INTERVIEWS

POST /applications/{id}/interviews

Schedule interview.

Request

```
{
  "mode": "video",
  "datetime": "2025-10-18T10:00:00Z",
  "link": "https://meet.medex.health/sara-apollo"
}
```

Response

```
{
  "id": "int_12d90e",
```

```
  "status": "scheduled"
}
```

Errors

```
INVALID_DATETIME

APPLICATION_NOT_FOUND
```

7.8 PAYMENTS & PLANS

GET /plans

List subscription plans.

Response

```
[
  {
    "id": "basic",
    "price": 999,
    "features": ["Post up to 5 jobs", "Email support"]
  },
  {
    "id": "pro",
    "price": 2999,
    "features": ["Post 25 jobs", "Priority listing", "Dedicated support"]
  }
]
```

POST /payments/subscribe

Create subscription.

Request

```
{
  "plan_id": "pro",
  "payment_method": "razorpay",
  "transaction_id": "txn_92adf3"
}
```

Response

```
{
  "subscription_id": "sub_445a9d",
  "status": "active",
  "expires_at": "2026-10-12T00:00:00Z"
}
```

7.9 ADMIN MODULE

GET /admin/users

Paginated user list.

Response

```
{
  "page": 1,
  "total": 1223,
  "data": [
    {
      "id": "b73a1c92-e6b2-41af-b1e1-04c93a87dfc8",
      "email": "dr.sara@example.com",
      "role": "candidate",
      "status": "active"
    }
  ]
}
```

PATCH /admin/jobs/{id}/moderate

Moderate job posts.

Request

```
{
  "status": "rejected",
  "reason": "Incomplete hospital license"
}
```

7.10 ERROR CODES & RESPONSES

Code	HTTP	Message	Context
AUTH_FAILED	401	Invalid credentials	Login
EMAIL_ALREADY_EXISTS	409	Email already registered	Signup
RESOURCE_NOT_FOUND	404	Resource not found	Jobs, Users
VALIDATION_ERROR	422	Invalid field	Input
INSUFFICIENT_PERMISSIONS	403	Role not allowed	HR-only endpoint
INTERNAL_SERVER_ERROR	500	Something went wrong	Fallback
PAYMENT_FAILED	402	Payment gateway declined	Razorpay/Stripe
RATE_LIMIT_EXCEEDED	429	Too many requests	Throttling

7.11 GRAPHQL (optional hybrid)

Endpoint: `/graphql`

Example query:

```
query CandidateDashboard($id: ID!) {
  profile(id: $id) {
    full_name
    specialization
    applications(limit: 5) {
```

```
      job {
        title
        location
      }
      status
    }
  }
}
```

7.12 SECURITY LAYER

JWT + Refresh token rotation every 24h

IP + Device fingerprint validation

Role-based access control via middleware

API throttling (100 req/min per user)

Audit log of all write operations

7.13 API Lifecycle & Docs

Documented via **OpenAPI 3.1 (Swagger)**

Sandbox: `https://sandbox.api.medex.health/v1/`

SDKs: auto-generated in `medex-js`, `medex-python`

API changelog maintained in `/v1/changelog`

---

8. SEARCH ARCHITECTURE & INDEXING STRATEGY

8.1 Core Goals of Search

Medex search must deliver:

Precision: relevant medical jobs per specialty, location, and qualifications.

Speed: < 200ms average query response time.

Personalization: candidate preferences, past searches, and specialty-weighted results.

Scalability: capable of handling 10M+ documents (jobs, profiles).

Resilience: indexing pipeline that can rebuild automatically after failure.

8.2 Components Overview

Component	Role	Technology
Search API layer	Handles query parsing, ranking, pagination	Node.js (NestJS) + Elasticsearch client
Indexing pipeline	Transforms DB data → searchable documents	Kafka / RabbitMQ + Worker (Node.js/Go)
Search engine	Core text + filter engine	Elasticsearch / OpenSearch
Vector service	Semantic matching	Sentence Transformers (BERT-based)
Cache layer	Stores frequent queries	Redis
Monitoring	Query latency & error logs	Kibana + Grafana

8.3 Data Sources & Index Types

Index	Purpose	Source Table(s)
jobs_index	All active jobs (private + govt)	jobs, employers, departments
profiles_index	Candidate searchable data	profiles, users, licenses
employers_index	Hospitals/clinics	employers, subscriptions
news_index	Medical job news & announcements	news_articles
govt_jobs_index	Special subset of govt jobs (pinned & vetted)	govt_jobs

---

8.4 Index Mappings (Elasticsearch example)

jobs_index

```
{
  "mappings": {
    "properties": {
      "job_id": { "type": "keyword" },
      "title": { "type": "text", "analyzer": "english" },
      "specialization": { "type": "keyword" },
      "description": { "type": "text", "analyzer": "english" },
      "location": { "type": "geo_point" },
      "city": { "type": "keyword" },
      "country": { "type": "keyword" },
      "salary_min": { "type": "integer" },
      "salary_max": { "type": "integer" },
      "experience_years": { "type": "integer" },
      "employment_type": { "type": "keyword" },
      "posted_at": { "type": "date" },
      "hospital_name": { "type": "text", "analyzer": "standard" },
      "is_government": { "type": "boolean" },
      "vector_embedding": { "type": "dense_vector", "dims": 384 }
    }
  }
}
```

---

profiles_index

```
{
  "mappings": {
    "properties": {
      "profile_id": { "type": "keyword" },
      "full_name": { "type": "text", "analyzer": "standard" },
      "specialization": { "type": "keyword" },
      "skills": { "type": "text" },
      "experience_years": { "type": "integer" },
      "location": { "type": "geo_point" },
      "preferred_cities": { "type": "keyword" },
      "availability": { "type": "keyword" },
      "vector_embedding": { "type": "dense_vector", "dims": 384 }
    }
  }
}
```

8.5 Indexing Pipeline (Flow)

Event trigger:

A job or profile is created/updated/deleted in PostgreSQL.

Event push:

A change event is sent to **Kafka** or **RabbitMQ**.

Worker processing:

Worker fetches the record, normalizes it, adds derived data (geo-coordinates, synonyms, embeddings).

Embedding service:

Title + description → sent to a BERT-based model to generate vector.

Bulk index:

Document is inserted or updated in Elasticsearch.

Confirmation & monitoring:

Indexing status logged; retries for failures.

Real-time sync latency:

Under 3 seconds per update.

8.6 Search Query Flow

Candidate Searching for a Job

Candidate inputs query `"pediatrician Jaipur government"`.

Query preprocessing:

Extract entities (`specialization=pediatrics`, `city=Jaipur`, `is_government=true`).

Spell correction + synonym expansion.

Hybrid search:

BM25 keyword relevance + vector semantic similarity.

Apply filters:

Salary, experience, job type.

Rank & personalize:

Weight based on user's past clicks, location proximity, and subscription jobs priority.

Return paginated JSON results (10–20ms retrieval time).

Employer Searching for Candidates

Employer queries `"MBBS doctor anesthesia 2+ years"`.

Similar NLP preprocessing.

Retrieve candidate profiles ranked by specialization, years, and geo-distance.

Boost verified and recently active profiles.

8.7 Advanced Features

Feature	Description
Synonym dictionary	“Obstetrician” ≈ “Gynecologist”; “Pediatrician” ≈ “Child Specialist”
Autocomplete	Prefix-based + popularity-ranked
Spelling correction	Levenshtein distance correction
Faceted filters	By specialty, location, salary range, job type, experience
Geo-distance filtering	Radius search (geo_distance: “within 50 km of Jaipur”)
Boosting rules	Featured jobs (+20%), new jobs (<7 days old)
Personalized ranking	Based on previous search and click history
Federated search	Unified results across jobs_index + govt_jobs_index
Vector reranking	Dense embeddings (BERT, MiniLM) used for semantic boost

---

8.8 Search Optimization Practices

Use **asynchronous indexing** to avoid blocking writes.

Refresh interval = 30s (not per-document).

Compress indices (best_compression codec).

Use **Elasticsearch aliases** for zero-downtime reindexing.

Maintain warm nodes with SSDs for top-queried regions.

Cache top queries in Redis with expiry 5m.

Monitor:

Query latency < 200ms p95.

Failed index events < 0.01%.

Node heap usage < 70%.

8.9 Security & Privacy in Search

Candidate anonymity: hide full name and contact unless candidate opts in.

Index only **hashed** identifiers, never raw emails or phone numbers.

GDPR-like compliance: right-to-delete triggers index document purge.

Audit all search queries made by employers.

Limit candidate searches for non-paying HR to 5/day.

8.10 Future Enhancements

LLM-based semantic matching: Use OpenAI embeddings or Sentence Transformers for better cross-specialty matching.

Query intent classification: “Government pediatrician jobs” → auto-route to govt index.

Dynamic reranking: Online learning-to-rank models using CTR and feedback loops.

Federated hybrid search: unify govt + private job search with distinct visual badges.

Multilingual search: use transliteration + multilingual models for Hindi/English input.

8.11 Example Search API Request

Endpoint: `/search/jobs`

Request

```
{
  "query": "pediatrician jaipur government",
  "filters": {
    "experience_min": 2,
    "salary_min": 50000,
    "job_type": ["full-time"]
  },
  "sort": "relevance",
  "page": 1,
  "limit": 10
}
```

Response

```
{
  "total": 42,
  "results": [
    {
      "job_id": "job_7d21e1c4",
      "title": "Pediatrician - Government Hospital",
      "hospital": "SMS Hospital Jaipur",
      "location": "Jaipur",
      "salary_range": "₹60,000 - ₹90,000",
```

```
      "distance_km": 3.2,
      "relevance_score": 0.94
    }
  ]
}
```

8.12 Metrics & Monitoring Dashboard

Metric	Target	Source
Avg query time	< 200 ms	Elasticsearch
Indexing delay	< 3 s	Kafka lag
Search success rate	99.9%	API logs
Top search terms	Pediatrics, Radiology, Government	Analytics
Candidate search frequency	≤ 5 per HR/day (free)	RBAC logs

---

9. Authentication, Authorization & RBAC (Role-Based Access Control) Matrix

Overview

Medex will employ a **multi-tier authentication system** and **granular authorization policy** to ensure secure access management across its three core user types — *Job Seekers*, *HR Recruiters*, and *Admins*. Security is a top priority due to sensitive user data (medical credentials, personal information, and employment records). The system will comply with **OWASP**, **HIPAA**, and **GDPR**-inspired best practices.

---

1. Authentication Architecture

a. Authentication Mechanism

Primary method: JWT-based token authentication (short-lived access token + refresh token).

Secondary method: OAuth 2.0 support (Google, LinkedIn, NMC/State Medical Council integration in future).

Optional 2FA: via SMS or authenticator app for Admins and HR accounts.

Password Hashing: Argon2 or bcrypt with salt.

Token Revocation: Stored in Redis or PostgreSQL blacklist table.

Session Timeout: 15 minutes idle timeout for HR/Admin; 30 minutes for Job Seekers.

b. Login Flow

User submits credentials or social login.

System verifies via `/api/auth/login`.

Generates Access Token (15 min) + Refresh Token (30 days).

Refresh tokens can be invalidated on logout or admin action.

Role and permissions are embedded in JWT payload (claims).

c. Password Recovery & Account Lock

Forgot password → OTP to verified email/mobile.

3 failed logins → soft lock for 10 minutes.

Admin override option for unlocking HR/Admin accounts.

2. Authorization Model

Medex employs **RBAC (Role-Based Access Control)** with **Permission-based Fine Control (PBAC)** overlay for sensitive features.
Each user role has a defined **scope**, **data visibility**, and **action privileges**.

---

3. Role Matrix

Role	Key Capabilities	Restricted Areas	Data Visibility	Notes
Job Seeker (User)	View & filter jobs, apply to jobs, manage profile/resume, track applications, receive notifications.	HR dashboard, Admin functions, payment sections.	Own data, job postings (public), HR limited view.	Can report HR abuse or fake postings.
HR Recruiter (Approved)	Post & manage jobs, view applicants, contact candidates, view analytics, manage subscription plans.	Admin panel, other HR's data, system settings.	Own postings, applicants for those jobs.	Must have active paid plan; suspended on expiry.
Admin (Super Admin)	Approve HRs, manage all users/jobs, view analytics, manage payments, post govt jobs, manage news, ban users.	None	Full system access	Has multi-admin support with granular permissions.
Moderator (Sub-admin)	Manage govt jobs, news, basic support tickets, verify reports.	HR payment & core settings.	Partial system view.	Delegated authority; optional.
System (Automation Services)	Background sync, job expiry, subscription reminders, analytics aggregation.	All user-facing UI.	DB-level limited scope via service account.	Runs via cron jobs or message queue workers.

---

4. Permission Set Examples

Permission Key	Description	Assigned To
job.create	Create new job posting	HR, Admin
job.delete	Delete job	HR (own), Admin
job.edit	Edit job posting	HR (own), Admin
job.view_all	View all job postings	Admin
user.manage	Manage user accounts	Admin

Permission Key	Description	Assigned To
govtjob.manage	Add/edit/delete govt jobs	Admin, Moderator
payment.manage	Manage payment plans & invoices	Admin
report.resolve	Handle abuse reports	Admin, Moderator
analytics.view	View system analytics	Admin
hr.approve	Approve or reject HRs	Admin
subscription.renew	Renew plan	HR, Admin
resume.view	View applicant resumes	HR (own job applicants), Admin
news.manage	Manage medical news content	Admin, Moderator

---

5. Access Control Middleware (Node.js / Express Example)

```
function authorize(requiredRoles = []) {
  return (req, res, next) => {
    const user = req.user;
    if (!user) return res.status(401).json({ message: "Unauthorized" });

    const hasRole = requiredRoles.includes(user.role);
    if (!hasRole) return res.status(403).json({ message: "Forbidden" });

    next();
  };
}

// Usage
app.post("/api/job", authorize(["HR", "Admin"]), createJob);
app.get("/api/admin/users", authorize(["Admin"]), listUsers);
```

---

6. Security & Compliance Enhancements

Area	Best Practice / Standard
Password storage	Argon2 / bcrypt with 12+ salt rounds
Encryption	AES-256 for sensitive PII (email, phone), TLS 1.3 for data in transit
Audit Trails	All CRUD actions logged with timestamp, IP, device info
API Rate Limiting	100 requests/minute per user
Account Verification	Email + Mobile OTP verification mandatory for HR/Admin
Session Hijack Prevention	SameSite cookies, CSRF tokens for sensitive forms
Log Monitoring	Security event logs stored in AWS CloudWatch / ELK stack
Compliance Goals	GDPR-like consent, HIPAA-aligned privacy handling, PCI-DSS for payments
Data Retention Policy	Auto anonymize inactive user data after 24 months

---

7. Token Schema (JWT Payload Example)

```
{
```

```
  "sub": "user_9853",
  "role": "HR",
  "permissions": ["job.create", "job.edit", "resume.view"],
  "exp": 1734028800,
  "plan": "6_months",
  "status": "active"
}
```

---

8. Admin Oversight & Escalation

Super Admin can override any access via dashboard.

Activity logs allow tracking of every admin/HR action.

Audit alerts for unauthorized access attempts.

---

10. Payments, Billing & Subscription Lifecycle

The **Medex** platform's financial backbone supports multiple subscription tiers, billing cycles, gateways, and automated lifecycle management for **HR Recruiters** (job providers). This ensures smooth monetization, transparent invoicing, and strict compliance with financial and security standards.

---

1. Payment & Subscription Overview

Purpose

To enable **HR recruiters** (hospital HRs, clinics, medical institutions, etc.) to:

Subscribe to **premium plans** (monthly to lifetime).

Post, manage, and promote job listings based on active plans.

Automatically renew, upgrade, or downgrade subscriptions.

Receive invoices, reminders, and payment history transparently.

Core Principles

Reliability – Payments must never fail silently.

Security – PCI-DSS compliance, HTTPS enforced, AES-256 encrypted storage.

Automation – Lifecycle events handled via background jobs (renewal, expiry, reminders).

Transparency – Real-time dashboard for HR and Admin to track payment status.

---

2. Subscription Plans

Plan Name	Duration	Max Job Posts	Features	Renewal	Price (example)
Basic	1 Month	5	Basic visibility, standard listing	Manual	₹999
Standard	3 Months	15	Featured in search, analytics access	Manual / Auto	₹2499
Professional	6 Months	30	Email alerts to users, highlighted jobs	Auto	₹4499
Premium	1 Year	75	Premium placement, analytics, contact	Auto	₹7499

Plan Name	Duration	Max Job Posts	Features	Renewal	Price (example)
			candidates		
Enterprise	2 Years	150	Dedicated HR dashboard support, bulk posting	Auto	₹12,999
Corporate	3 Years	300	Priority support, job performance analytics	Auto	₹19,999
Lifetime	Lifetime	Unlimited	Permanent posting rights, unlimited data access	N/A	₹49,999

(Pricing is indicative; can vary by market or promo code.)

---

3. Payment Gateway Integration

Supported Gateways

Primary: Razorpay / Stripe / PayPal (for global users).

Backup: Paytm / PhonePe Business (for Indian domestic payments).

Integration Flow

HR selects a plan → `/api/payments/checkout`.

API generates an **order_id** with the payment gateway.

User completes payment → Gateway sends webhook to `/api/payments/webhook`.

Payment verified → Subscription record created or updated.

Invoice generated & emailed to HR and stored in DB.

Dashboard reflects new expiry date and job limits.

---

4. Subscription Lifecycle Stages

Stage	Trigger	Action	System Event / Job
Trial (optional)	Signup	Limited posting (1 job)	Auto expire after 7 days
Active	Successful payment	HR can post/manage jobs	Allow job posting
Expiring Soon	7 days before expiry	Reminder email/SMS	cron_notify_expiry()
Expired	Plan end date passed	Job posting disabled	cron_deactivate_plan()

Stage	Trigger	Action	System Event / Job
Grace Period	7 days post-expiry	Temporary hold	grace_mode=true
Renewed	Payment received before expiry	Extend expiry date	updateSubscription()
Cancelled	Manual or failed payment	Suspend account	markInactive()
Refunded	Manual admin approval	Reverse payment	Gateway API call
Upgrade/Downgrade	New plan chosen	Prorate balance	calculateProrate()

---

5. Payment API Endpoints

Method	Endpoint	Description
POST	/api/payments/checkout	Create payment order
POST	/api/payments/webhook	Receive gateway event (success/failure/refund)
GET	/api/payments/subscription	View active plan and expiry
POST	/api/payments/renew	Renew existing plan
POST	/api/payments/upgrade	Upgrade/downgrade plan
GET	/api/payments/invoices	List all invoices
GET	/api/payments/invoice/:id	Get specific invoice PDF
POST	/api/payments/refund	Request refund (Admin approval needed)

---

6. Payment Data Schema

Table: **payments**

```
CREATE TABLE payments (
  id SERIAL PRIMARY KEY,
  hr_id INT REFERENCES hr_users(id),
  plan_id INT REFERENCES plans(id),
  amount DECIMAL(10,2),
  currency VARCHAR(10) DEFAULT 'INR',
  status VARCHAR(20) CHECK (status IN ('pending','success','failed','refunded')),
  transaction_id VARCHAR(255),
  order_id VARCHAR(255),
  gateway VARCHAR(50),
  payment_date TIMESTAMP DEFAULT NOW(),
  expiry_date TIMESTAMP,
  invoice_url TEXT,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);
```

Table: **plans**

```
CREATE TABLE plans (
  id SERIAL PRIMARY KEY,
  name VARCHAR(50),
  duration_months INT,
  price DECIMAL(10,2),
```

```
  max_jobs INT,
  features JSONB,
  created_at TIMESTAMP DEFAULT NOW()
);
```

---

7. Invoice & Billing Management

Automated Invoicing System

Each transaction generates a unique **invoice number**.

PDF invoice auto-generated (with GST/Tax fields).

Delivered via email + downloadable from dashboard.

Stored securely in AWS S3.

Invoice Fields

Invoice ID, Date, Buyer Info (HR details), Plan Name, Price, Tax, Total, Payment Status, Signature (Digital).

Integration: `pdfkit` or `jsPDF` for Node.js + AWS SES for delivery.

---

8. Refund & Dispute Resolution

Refund requests via dashboard → reviewed by Admin.

Only unused or failed payment cases qualify.

Refunds processed via original payment gateway (API call).

Transaction and refund logs kept for audit trail.

Dispute escalation email chain: support@medex.in.

---

9. Automated Reminder & Renewal Jobs

Job Name	Frequency	Purpose
cron_notify_expiry()	Daily	Remind HR of expiring plans
cron_suspend_expired()	Hourly	Disable expired HR accounts
cron_autorenew()	Daily	Auto-renew active cards (Stripe etc.)
cron_invoice_archive()	Weekly	Archive invoices to S3/Cold Storage

---

10. Security & Compliance

Standard	Implementation
PCI-DSS	No card data stored on Medex servers
TLS 1.3	Enforced across all endpoints
Webhook Signature Verification	SHA256 HMAC validation
Encryption	AES-256 for transaction IDs, bcrypt for sensitive fields
Fraud Detection	Geo-IP & velocity checks on multiple payments
Audit Trail	Immutable logs for all billing actions
Data Retention	7 years for audit; anonymized thereafter
GDPR Compliance	Consent for recurring billing required

---

11. Admin Billing Dashboard

Features:

View all transactions by date range, HR, status.

Approve refunds or cancellations.

Generate revenue analytics charts.

Export reports (CSV/XLSX/PDF).

Manage plans and pricing dynamically.

Monitor gateway uptime and webhook logs.

---

12. Future Extensions

Promo codes & coupons (percentage/flat).

Corporate accounts with multiple HR logins.

Revenue sharing model with medical associations.

Crypto or UPI AutoPay integration for emerging markets.

AI-based churn prediction from payment activity

---

11. Verification, Fraud Detection & Moderation Rules — Detailed Design

This section defines how **Medex** will verify users and employers, detect and prevent fraud, and moderate content. It covers verification workflows, automated heuristics, scoring & thresholds, admin tooling & playbooks, appeals, audit trails, metrics to monitor, sample API surfaces, privacy/legal considerations, and implementation recommendations (including ML where useful).

Goals & Principles

Trust: Ensure that employers and candidates are who they say they are (medical councils, hospital HRs).

Safety: Prevent scams, fake jobs, phishing, and data harvesting.

Usability: Minimize friction for legitimate users — combine automated checks with human reviews.

Transparency & Fairness: Provide clear reasons for rejections and an appeals mechanism.

Traceability & Auditability: Log all decisions and retain records for compliance and dispute resolution.

---

1 — Verification Types & Badges

Candidate verifications

Email Verified — automatic OTP/email confirmation (badge: ✔ Email).

Phone Verified — OTP SMS confirmation (badge: ✔ Phone).

Medical Registration Verified — validate medical license/registration number against council where possible (badge: ✔ Registered).

Methods: API lookups (where councils provide APIs), manual admin verification (documents), or third-party verification partner.

Identity Verified (optional) — government ID (Aadhaar/Passport) check (manual or third-party).

Background-checked (optional paid add-on) — third-party background screening (badge).

Employer (HR / Organization) verifications

Domain Verified — employer email domain matches website domain and DNS check passes (badge).

KYC Verified — company registration docs (GST/PAN/company certificate) manually reviewed and approved by admin (badge: ✔ KYC).

Trusted Employer — longer-standing verified employers with positive history get elevated badge (tiered).

Payment Verified — employer has valid subscription & payment history (displayed in dashboard).

---

2 — Verification Workflows

Candidate — Medical Registration Verification (recommended flow)

Candidate supplies registration number, issuing authority, and uploads certificate image.

System attempts **automated lookup**:

If council API exists → query and validate (status, name match).

If not available → perform heuristic checks (format, cross-check name + DOB).

If automated check fails or is inconclusive → enqueue for **manual review** by admin.

Admin reviews docs, may ask candidate for additional proof; then marks `verified` or `rejected` with reasons.

On `verified`: award badge on profile and expose to employers (per privacy settings).

On `rejected`: notify candidate with clear reasons and remediation steps.

Timing SLAs: Automated: <10s. Manual: target <72 hours (SLA configurable).

Employer — KYC Verification

Employer registers with org email; uploads company registration docs + authorized signatory ID + website link.

System runs **automated checks**:

Domain ownership (MX records, SPF, DMARC), WHOIS check, website existence, reverse email domain.

Phone number verification via OTP.

Quick risk checks: newly-created domain, disposable email, geographic mismatch.

If auto-check passes and risk score low → provisional approve (badge `KYC: pending manual` or auto-approve per policy).

If risk score moderate/high → manual admin review: check docs, call POC, check official listings.

On approval → `kyc_status = approved` and HR allowed to purchase plans; otherwise `rejected` with reasons.

SLA: Aim for manual review <48–72 hours.

3 — Fraud Detection System (FDS)

A layered system combining **rules-based heuristics**, **behavioral signals**, and **ML models**.

3.1 Data inputs for fraud scoring

Account metadata: email domain, registration age, phone carrier, IP address, device fingerprint.

Behavioral signals: posting frequency, time between signup → first job, number of unique jobs posted in short window, job duplicates, text similarity across postings.

Payment & billing: chargeback history, refund frequency, failed payments.

Reputation signals: number of verified hires, previous appeals, user reports.

External signals: WHOIS age, social presence, public complaints.

3.2 Heuristics & Rule Examples

Disposable email / free domain: +30 risk.

New domain (<30 days): +20 risk.

Multiple jobs from same IP in <1 hour: +25 risk.

Phone number mismatch to country: +15 risk.

High chargeback ratio (>5%): +50 risk.

Posting contact details in job description (bypass platform): +40 risk (auto-hide and flag).

Jobs with salary = 0 or "pay after interview": +20 risk.

Repeated removals for policy violations: +60 risk.

3.3 ML Enhancements (Phase 2)

Anomaly detection using unsupervised models (isolation forest) to catch patterns unseen by rules.

Text classifiers to detect scam-like language (requests for money, personal bank info, vague benefits).

Behavioral models to predict likelihood of employer being fraudulent based on historical features.

Ranking model to surface most likely legitimate candidates/employers for human review.

3.4 Scoring & Thresholds

Maintain a numeric **risk score** (0–100).

Policies:

`risk < 30` → allow auto-approve (low risk).

`30 <= risk < 60` → soft flag (allow but monitor + limit features).

`risk >= 60` → block posting & require manual review (suspend until resolved).

Thresholds configurable by admin and should be tuned using historical data.

---

4 — Moderation Rules & Content Policies

What is disallowed (hard rules)

Job posts asking for **upfront payments** from candidates.

Posts soliciting **sensitive personal data** (bank account, Aadhaar number) in job descriptions.

Posts that request or encourage illegal activity.

Pornographic or sexually explicit content.

Hate speech, discrimination (race, religion, sexual orientation) or illegal discrimination.

Phishing links or external sites that attempt to harvest PII.

Soft rules (flag for review)

Extremely vague job descriptions (e.g., “Urgent hiring — contact ASAP”) — flag for minimal info.

Salary extremely below market median — flag for potential bait.

Non-medical roles posted in medical categories — flag for miscategorization.

Automatic mitigations

Auto-strip/obfuscate phone numbers or email addresses in job descriptions and replace with “Contact via platform”. Flag the posting for admin review.

Auto-hide posts with high-risk terms until human review passes.

---

5 — Admin Moderation Tools & Workflows

Admin dashboard features

KYC queue with filters (risk_score, submitted_at, country).

Flagged content queue (jobs, messages, resumes).

Fraud dashboard: top flagged employers, trends, manual override buttons (suspend/unblock).

Investigation pane: view account history, IP addresses, device fingerprints, payment history, text similarity matches, prior complaints.

Action buttons: Approve, Reject (with template reasons), Suspend (temporary), Ban (permanent), Request More Documents.

Appeals management with status tracking and response templates.

Playbook examples

High risk employer (>=60): Auto-suspend; prioritize manual review within 24h; block posting; notify account owner with required docs.

Fake job detected: Pause job posting, notify candidates who applied with safe advisory, list remediation steps and open investigation ticket.

Candidate reports employer: Create support ticket & escalate to fraud team; if claim credible, suspend employer pending review.

---

6 — Appeals & Dispute Resolution

Candidate/employer-facing appeals

When an action (suspend/reject) occurs, send clear notification: reason code, evidence summary, what documents are needed, and link to appeal form.

Appeal form fields: account id, incident id, attach documents, contact info.

SLA: acknowledge appeal within 24 hours; resolve within 72 hours (target).

Internal adjudication

Triage team reviews appeal with audit logs & evidence.

Possible outcomes: Restore Account, Partial Restore (limited features), Uphold Suspension. All actions logged.

---

7 — Messaging & Notifications (Templates & Tone)

Suspension notice (example)

```
Subject: Action Required — Employer Account Temporarily Suspended
Body: We detected activity that requires verification. To restore posting
rights please upload [company registration document] and [authorized
signatory ID]. Your account will remain suspended until verification
completes.
```

Job removed notice (candidate-visible)

```
Subject: Job Removed — [Job Title]
Body: We removed this job due to a policy violation. If you had applied, we've notified the employer and will notify you of any updates. Contact support for questions.
```

---

8 — Logging, Audit Trails & Evidence Preservation

Audit logs for every moderation decision (actor_id, action, reason_code, metadata, timestamp).

Immutable evidence store: store snapshots of job content, KYC docs, communication transcripts before taking action (retention per legal policy).

Chain-of-custody: record who accessed evidence, when, and what actions were taken. Valuable for legal disputes.

---

9 — APIs & Automation Surfaces

POST /fraud/flag — programmatic flag. Body: `{ entity_type: 'employer'|'job'|'user', entity_id, reason_codes: [], metadata }`

GET /admin/kyc-queue?risk_gt=30 — paginated KYC queue for review.

PATCH /admin/employers/{id}/suspend — suspend account with reason.

POST /admin/appeals/{id}/resolve — resolve an appeal and annotate.

All admin APIs gated with `admin` RBAC and require 2FA for high-risk actions.

---

10 — Metrics & Monitoring

Track both operational health and accuracy of fraud/moderation:

Operational metrics

KYC throughput: avg time from submission → decision. (Target: <72h).

Flags per day, actions per day, appeals open/closed.

Number of suspended accounts and reinstatements.

Quality metrics

True positive rate (TPR): proportion of flagged accounts truly fraudulent (measured by manual review outcomes).

False positive rate (FPR): flagged but legitimate — track appeals overturn rate.

Time to resolution (median).

Candidate impact: % of applicants affected by job removals.

Set dashboards and alerts for spikes (sudden increase in flags, chargebacks, or complaints).

---

11 — Privacy & Legal Considerations

Data minimization: Only store needed PII and keep it encrypted.

Retention & legal hold: KYC evidence kept for required time; legal hold can prevent automatic deletion.

Transparency: Publish clear policies about verification and fraud prevention in T&Cs and Privacy Policy.

Cross-border data flows: If verifying against external authorities, ensure consent and lawful basis for cross-border transfers.

Reporting obligations: For severe criminal activity, provide law enforcement with evidence per legal process; have SOPs and designated legal contacts.

---

12 — Implementation Roadmap & Priorities

Phase 1 (MVP — required)

KYC upload & admin review flow.

Basic heuristics rules (disposable email, new domain, phone OTP).

Flagging UI & manual suspend/approve actions.

Audit logs and notification templates.

Phase 2 (improve)

Add more heuristics, automated domain/WHOIS checks, rate-limiters.

Basic text classifiers (scam detection) using open-source models.

Auto-obfuscation of contact details and auto-hide of suspicious posts.

Phase 3 (advanced)

ML anomaly detection & scoring models; online learning with feedback loop.

Third-party verification integrations (medical council APIs, background-check providers).

Automated risk-based workflows (e.g., auto-approve low-risk, auto-suspend high-risk).

---

13 — Example Risk Scoring Pseudocode (simplified)

```
score = 0
if email_domain in disposable_domains: score += 30
if domain_age_days < 30: score += 20
if posting_count_last_hour > 10: score += 25
if phone_geo_mismatch: score += 15
if text_contains("pay", "deposit", "wire"): score += 40
if chargeback_rate > 0.05: score += 60

# normalize and return
risk_score = min(score, 100)
```

Tune weights using labeled historical data.

---

14 — Example Moderation Decision Matrix

Risk Score	Auto-action	Admin action required
0 – 29	Allow	No
30 – 59	Allow with restrictions (limit job reach, monitor)	Optional review
60 – 79	Block posting; temp suspend	Manual review within 24 – 48h
80 – 100	Immediate ban & legal review if evidence of fraud	High priority manual review

---

15 — Operational Playbook Snippets (for moderation team)

Case: Employer posts jobs with contact details bypassing platform

Auto-hide contact info, flag as high risk, notify admin.

Send email to employer: “Posting with external contacts is against policy — please edit within 24 hours or account will be suspended.”

If no remediation: suspend and open appeal ticket.

Case: Candidate reports harassment from employer

Temporarily suspend employer posting rights.

Collect full evidence (screenshots, message logs), notify both parties of temporary action.

Escalate to legal if necessary.

16 — Wrap-up & Next Steps

Implement MVP heuristics & admin UI — get initial protection quickly.

Instrument everything (flags, appeals, actions) to produce labeled data for ML.

Pilot ML models on historical data and run as advisory before moving to automated blocking.

Draft public policies & clear communication templates for users and employers.

Regular reviews — tune thresholds monthly in the first 6 months.

12. Security & Compliance Controls

Medex handles sensitive professional data from doctors, paramedical staff, and healthcare organizations, requiring **enterprise-grade security**, **regulatory alignment**, and **robust monitoring**. This section defines the framework for protecting data, ensuring compliance, and maintaining system integrity under Indian and international standards (HIPAA, GDPR, ISO 27001).

---

1. Security Governance Framework

Layer	Focus Area	Standards / Practices Followed
Governance	Policies, roles, and oversight	ISO 27001 / NIST Cybersecurity Framework
Application Security	Secure coding, API hardening	OWASP Top 10, CWE/SANS
Data Security	Encryption, anonymization	HIPAA, GDPR (Articles 32–35)
Infrastructure Security	Cloud access, firewalls, network segmentation	CIS Benchmarks, Zero Trust Architecture
Incident Response	Threat monitoring and escalation	NIST 800-61
Business Continuity	Backups and DR readiness	ISO 22301

---

2. Data Security Controls

a. Encryption

At Rest: AES-256 encryption for all PII, documents (CVs, certificates).

In Transit: TLS 1.3 for all HTTPS traffic.

Field-Level Encryption: Sensitive identifiers (Aadhaar, PAN, Medical Council Number) stored encrypted separately.

Key Management: AWS KMS or Vault, rotated every 90 days.

b. Data Classification

Data Type	Sensitivity Level	Handling Policy
Name, email, phone	Medium	Encrypted at rest
Medical registration number, certificate scans	High	Encrypted + access logged
Payment details	High	Tokenized, PCI-DSS compliant
Job listings, news	Low	Public access via filtered API

c. Data Retention & Deletion

Inactive user data anonymized after **24 months**.

Account deletion removes identifiable data, retaining metadata for audit (7 years max).

Job application data retained for **180 days post-closing**.

---

3. Infrastructure & Network Security

a. Cloud Setup (AWS / GCP)

VPC segmentation: separate subnets for **App**, **DB**, **Admin**, and **Monitoring** layers.

WAF (Web Application Firewall) protection via AWS Shield / Cloudflare.

Security groups whitelist ports (80, 443, 22 restricted to admin IPs).

Bastion host for SSH access only via VPN.

Continuous vulnerability scanning (AWS Inspector / Prisma Cloud).

b. Database Security

Role-based DB access with least privilege principle.

No direct internet-facing DBs.

Encryption keys stored separately from database instance.

Query logs monitored for anomaly detection.

c. File & Object Storage

CVs, certificates, and documents stored in **S3 buckets** with signed URL access only.

Antivirus scan for all uploaded files (ClamAV + Lambda trigger).

S3 bucket access logs monitored via CloudTrail.

---

4. Application Security Controls

a. Secure Development Lifecycle (SDLC）

Code review required before merge (GitHub Actions enforced).

Static code analysis (Snyk / SonarQube).

Dependency vulnerability scanning (npm audit CI).

Unit and penetration tests before deployment.

Environment secrets via Vault or environment variables (no plaintext config).

b. OWASP Protection

Threat	Mitigation
SQL Injection	ORM (Prisma/Sequelize) + parameterized queries
XSS	React auto-escaping + CSP headers
CSRF	Anti-CSRF tokens on all form submissions
Broken Auth	Short-lived tokens + refresh rotation
Insecure Direct Object Reference	Access control checks on all resource IDs
Misconfiguration	Auto security headers (Helmet.js), rate limiting

---

5. Monitoring & Incident Response

a. Logging & Auditing

Centralized logging via ELK / CloudWatch.

Every CRUD operation tagged with:

User ID

IP

Timestamp

Device Fingerprint

Action Type

Tamper-proof audit trail for 7 years.

b. Real-Time Security Monitoring

IDS/IPS integration (Snort / AWS GuardDuty).

Alert thresholds for login anomalies (multiple failed attempts, new device).

Automated notification to Admins for suspicious HR activity.

c. Incident Response Plan (IRP)

Detection → Immediate alert to security team.

Containment → Revoke tokens / suspend accounts.

Eradication → Fix vulnerability and re-scan.

Recovery → Restore from clean backup.

Post-Mortem → Document & improve.

MTTD (Mean Time to Detect): ≤15 min
MTTR (Mean Time to Respond): ≤2 hours

---

6. Compliance & Legal

Regulation / Framework	Applicability	Implementation Details
HIPAA (US)	Handling doctor/patient-related data	Encryption, access logging, BAAs if extended
GDPR (EU)	Users from EU region	Right to access/delete data, consent banner
DPDP Act (India 2023)	All Indian users	Data fiduciary compliance, user consent records
PCI-DSS	Payment gateway integration	Tokenized card data via Razorpay/Stripe
ISO 27001	Optional certification goal	Full ISMS documentation planned for enterprise scaling

---

7. Backup & Disaster Recovery (DR)

Frequency: Full DB backup every 6 hours; incremental every 15 minutes.

Storage: Multi-region replication (e.g., Mumbai + Singapore).

DR Target: RPO (Recovery Point Objective): ≤15 min ; **RTO (Recovery Time Objective):** ≤2 hours

Test Drills: Quarterly simulated outage recovery.

---

8. Privacy & Consent Management

Consent checkbox mandatory for all new users.

Consent version stored in DB with timestamp.

Privacy policy versioned and linked to consent record.

Cookie banner compliant with GDPR / DPDP.

Email marketing opt-in stored separately.

Data Subject Request (DSR) API for account deletion/export.

---

9. Security Testing & Auditing

Test Type	Frequency	Performed By
Static Analysis (SAST)	Every commit	Automated
Dynamic Analysis (DAST)	Every release	Security engineer
Penetration Test	Quarterly	External auditor
Compliance Audit	Annual	ISO-certified auditor

---

10. Security Metrics & KPIs

Metric	Target
Mean Time to Detect (MTTD)	≤ 15 min
Mean Time to Resolve (MTTR)	≤ 2 hours
Vulnerabilities fixed per month	≥ 95%
Patch compliance	100% within 7 days
Unauthorized access incidents	0 critical breaches/year

---

11. Security Culture & Training

Developer onboarding includes **OWASP Secure Coding** module.

HR/Admin users receive periodic **phishing awareness** training.

Quarterly “tabletop exercises” for incident response simulations.

Annual re-certification for Admins on data handling policies.

---

13. Privacy & Data Retention Policies

Medex, as a healthcare-oriented job portal handling sensitive personal and professional data, must uphold **strict privacy protections**, **ethical data handling**, and **transparent retention rules**. The following section details how user data is collected, processed, stored, shared, and deleted — in compliance with **India's DPDP Act (2023)**, **GDPR**, and **HIPAA-like principles** of confidentiality and minimal exposure.

---

1. Privacy Philosophy

Medex is built on three guiding principles:

Data Minimization: Collect only what's necessary to deliver platform functionality.

User Control: Allow users to view, edit, and delete their data anytime.

Transparency: Clearly inform users of data usage, sharing, and retention timelines.

Our commitment extends to all stakeholders — doctors, paramedical staff, AYUSH professionals, HR recruiters, and administrators.

---

2. Data Categories Collected

Category	Examples	Purpose	Retention Period	Access Level
Account Data	Name, email, mobile, password hash, role	Login & authentication	Until account deletion	User, Admin
Professional Data	Qualification, specialty, registration number, experience, license document	Resume building & verification	24 months post-inactivity or until deletion	User, HR (for applicants), Admin
Job Data	Job title, employer, salary, location, description	Listing and discovery	Until job expiry + 180 days	Public (filtered), Admin
Application Data	CV, cover letter, messages, applied jobs	Application tracking	180 days post job closure	HR (own postings), Admin
Billing Data	Plan type, invoice ID, payment status	Subscription management	7 years (tax compliance)	Admin, Finance
Device & Usage Data	IP, browser, device ID, activity logs	Security & analytics	12 months	Admin only

Category	Examples	Purpose	Retention Period	Access Level
Government Jobs Data	Notices, PDFs, application links	Public dissemination	Until job deadline + 180 days	Admin
Cookies & Tracking Data	Session ID, preferences, analytics events	Personalization & site analytics	6 months	Internal systems only

---

3. Consent & Legal Basis

Medex will collect and process data under these legal bases:

Consent (Primary): For account creation, email communication, marketing.

Contractual Necessity: For fulfilling platform services like job application or HR posting.

Legal Obligation: For retention of billing and financial records.

Legitimate Interest: For platform analytics, fraud prevention, and service improvement.

Consent Management Features:

Explicit opt-in checkbox during registration.

Separate consent for marketing emails.

Granular privacy dashboard to withdraw or modify consent.

Consent versioning (audit trail of user consents with timestamp and version number).

---

4. User Rights (in line with GDPR & DPDP)

Right	Description	How Implemented
Access	View what personal data Medex holds	Download profile & activity log
Rectification	Edit incorrect or outdated info	Profile editing tools
Erasure ("Right to be Forgotten")	Delete account & associated data	“Delete Account” button triggers soft delete → purge after 30 days
Data Portability	Export data in machine-readable format	JSON or CSV export from dashboard
Restriction of Processing	Pause account visibility without deletion	Privacy mode toggle

Right	Description	How Implemented
Objection to Processing	Opt-out from marketing or analytics	Preference center
Automated Decision Transparency	No fully automated rejections	HR reviews applications manually

All user requests are verified via email or OTP before execution to prevent unauthorized actions.

---

5. Data Retention Rules

a. Active Accounts

Data retained as long as the user maintains an active account.

Users can modify or delete information anytime.

b. Inactive Accounts

After **24 months of inactivity**, accounts enter archival state.

Data anonymized but recoverable on reactivation.

30-day notice sent before deletion.

c. Deleted Accounts

Soft delete (user-hidden) for 30 days.

Hard delete (complete erasure) after 30 days unless legally required otherwise.

Metadata (logs, audit trails) kept for **7 years** for compliance.

d. HR Data

HR subscription records retained 7 years for financial audits.

Job postings auto-deleted 180 days after closure.

Applicant resumes deleted after 180 days unless HR explicitly requests retention.

e. Government Job Data

Admin-posted government jobs automatically expire and are archived after 180 days.

---

6. Data Sharing & Disclosure

Scenario	Shared With	Purpose / Limitation
Job applications	HR Recruiters	Only for applied jobs; no bulk access
Payment processing	Razorpay / Stripe	Tokenized transaction processing
Legal compliance	Govt / regulatory authorities	Only under valid legal request
Analytics	Google Analytics (or self-hosted Plausible)	Anonymized usage data only
Marketing emails	SendGrid / Mailchimp	Opt-in users only
Cloud hosting	AWS / GCP	Data encrypted at rest & in transit

No data is ever sold or rented to third parties.

---

7. Privacy by Design (PbD) Implementation

Minimal Data Entry: Job seekers need only essential details initially.

Role-based Data Visibility: HRs can view only applicants for their jobs.

Anonymized Analytics: Site usage tracked without personally identifiable info.

Secure Defaults: All profiles private until user opts for public visibility.

Continuous Review: Annual privacy audits; external audits optional for ISO 27701 alignment.

---

8. Cookie Policy

Essential Cookies: Required for session & authentication.

Functional Cookies: Remember preferences and filters.

Analytics Cookies: Only loaded after consent.

Third-party Tracking: Disabled by default.

Cookie consent banner (multi-language) with “Accept All / Reject All / Customize” options.

---

9. Children & Restricted Users

Platform restricted to **18+ verified professionals**.

No data collected knowingly from minors.

Verification via professional ID or registration number.

---

10. Breach Notification Protocol

If a data breach occurs:

Notify affected users within **72 hours** (GDPR-compliant).

Report to relevant Indian Data Protection Board if DPDP Act applies.

Include details of nature, scope, and mitigation measures.

Provide credit monitoring or remedial support if applicable.

---

11. Data Anonymization & Aggregation

Before deletion, inactive user data is anonymized for analytics:

Remove identifiers (email, phone, ID).

Retain non-personal statistics (specialty, city, years of experience).

Use anonymized data for trend reporting and platform performance.

---

12. Third-Party Compliance Requirements

All partners, vendors, and payment processors must sign:

Data Processing Agreement (DPA).

Non-Disclosure Agreement (NDA).

Breach Reporting SLA (24 hours from discovery).

---

13. Personal Data Collected

Data Type	Collected From	Usage Purpose	Storage Duration	Encryption
Identity Data (Name, DOB, Gender)	Job Seekers, HR	Account creation, resume, job posting identity	7 years post inactivity	AES-256

Data Type	Collected From	Usage Purpose	Storage Duration	Encryption
Contact Info (Email, Phone, Address)	All users	Communication, verification, recovery	7 years post inactivity	AES-256
Professional Info (Qualifications, License, Reg. No., Experience)	Job Seekers, HR	Resume visibility, job matching, HR vetting	7 years	AES-256
Financial Data (Payments, Invoices)	HR	Subscription billing, tax compliance	10 years (per accounting laws)	AES-256
Device/IP/Log Data	All users	Security monitoring, fraud detection	18 months	SHA-256 hashed
Uploaded Docs (Certificates, CVs)	Job Seekers	Profile enrichment, admin verification	Until user deletion	Encrypted at rest (S3 AES-256-GCM)
Analytics/Usage Data	All users	Product improvement	Aggregated after 12 months	Pseudonymized

15. UX, Accessibility & SEO Specifications

This section gives practical, implementable rules and examples so designers, frontend engineers and content teams can ship an inclusive, high-converting Medex site that ranks well and performs reliably. It covers UX principles, UI components & behaviours, WCAG accessibility requirements (WCAG 2.1 AA), keyboard/ARIA patterns, and SEO/SSR rules including structured data and performance targets.

A — UX Principles & Design System (practical)

Core UX principles

Mobile-first, content-first. Design for smallest screens first and progressively enhance.

Trust-forward. Prominent verification badges, KYC states, and clear privacy controls.

Low-friction flows. Minimize typing: OTP, progressive profiles, pre-fill from resume parsing.

Task-focused pages. Each page solves one job: search, view job, apply, manage jobs.

Predictable patterns. Consistent placement of primary CTAs, search, filters and user menus.

Performance-aware interactions. Prefer skeleton loaders and optimistic updates.

Design system essentials

Color palette: high contrast primary (navy/teal) + neutral greys + accent for CTAs.

Scales: spacing (4,8,12,16...), typography scale (16px base), 2xl rounded corners on cards.

Tokens: central tokens for colors, spacing, radii, shadows.

Components: Button, Input, Select, Tag, JobCard, JobDetail, FilterPanel, Modal, Toast,Pipeline (Kanban), DataTable, KYCDocumentViewer, Avatar, Badge.

Microcopy: friendly, clinical-professional tone (clear CTAs, error language actionable).

Interaction patterns

Primary CTA (Apply / Post job) always visible on JobDetail.

Inline validation with helpful messages and ARIA `aria-invalid` + `aria-describedby`.

Progressive disclosure: show advanced filters only when expanded.

Empty states with clear next steps (e.g., “No jobs here — save a search” CTA).

Error pages: meaningful 404/500 pages with search box and contact link.

B — Component-level UX rules (developers)

Buttons

Use `<button>` elements for actions; provide `aria-label` when icon-only.

Disabled state visually distinct and `aria-disabled="true"`.

Forms

Place labels above inputs; use placeholders only for examples.

Required fields marked with `aria-required="true"` and `required` attribute.

Error messages inline + associated with input via `aria-describedby`.

Modals

Trap focus inside modal while open; restore focus to trigger on close.

Use `role="dialog"` and `aria-modal="true"`. Title connected via `aria-labelledby`.

Lists / JobCard

Provide keyboard-accessible list items (`role="list"`, job cards as buttons/links).

Include succinct metadata: specialty tags, location, salary range, posted date and verification badge.

Filters

Collapsible, persist state in URL (query params) for shareable links.

Use checkboxes & sliders with visible keyboard control.

Notifications

Toasts: unobtrusive; use `role="status"` for polite screen-reader announcements.

Critical alerts use `role="alert"`.

C — Accessibility (WCAG 2.1 AA) — must-have checklist

Per-page / site-level

Contrast: text ≥ 4.5:1 for normal text; large text ≥ 3:1.

Keyboard navigation: full functionality via keyboard (Tab, Shift+Tab, Enter, Space, Arrow keys).

Skip links: `Skip to main content` anchor at top of each page.

Focus styles: visible, high-contrast focus indicator for interactive elements.

Semantic HTML: headings (`h1`..`h6`), lists, tables used correctly.

Forms & inputs

Labels associated via `for` / `id`.

Errors announced to screen readers; use `aria-live="assertive"` for immediate errors.

File upload: provide progress and alternative method (email/manual) for failing uploads.

Images & icons

All informative images have `alt` text; decorative images `alt=""` and `aria-hidden="true"`.

Icons with only visual meaning must have `aria-hidden="true"`; icon buttons need `aria-label`.

Dynamic content

Use `aria-live` regions for async changes (apply confirmation, status updates).

For SPA updates, ensure screen readers are notified of significant navigation or content change.

Multimedia

Video interview pages require captions and transcripts.

Audio-only content must have transcript.

Testing

Automated: axe-core, pa11y in CI.

Manual: screen-reader walkthroughs (NVDA/VoiceOver), keyboard-only flows, color-blindness checks.

Acceptance: pass automated scan + 2 manual audits per release.

D — Internationalization & Localization

All strings externalized using i18n; use ICU formatting for plurals/dates.

Right-to-left support capability in the design system.

Locale-aware date, number, currency formatting.

Hreflang support for multi-language pages.

E — UX Metrics & Instrumentation

Track UX success via events and dashboards:

Funnel: `search.rendered` → `job.view` → `apply.started` → `apply.completed`.

Speed: time-to-interactive (TTI), First Contentful Paint (FCP), Largest Contentful Paint (LCP).

Engagement: saved searches, job alerts signups, resume uploads completed.

Error rates: form validation failures per form field.

Accessibility regressions: count of a11y violations per build.

Use analytics (Segment/PostHog) + custom dashboards (Metabase/Looker).

F — SEO Specifications (technical + content)

Goals

Maximize discoverability for medical job queries and government posts.

Ensure authoritative indexing of govt jobs and featured listings.

Page rendering

SSR for job pages and landing pages (Next.js getServerSideProps/ISR for near-real-time). Public job pages should be server-side rendered or statically generated and revalidated (ISR) to serve full HTML for crawlers.

Use `Link` prefetching for internal navigation.

Technical SEO basics

Unique, descriptive `<title>` and `<meta name="description">` per page (70 / 160 chars guidelines).

Canonical tags for duplicate content.

`rel="alternate" hreflang="x"` for multi-language sites.

Robots.txt: allow main site crawl; disallow admin, staging, private job preview urls.

Sitemap: `/sitemap.xml` updated daily for high-change pages (govt jobs). Include priority and changefreq.

Structured Data: use `JobPosting` JSON-LD for job pages and `BreadcrumbList` for nav. See example below.

Structured Data — JobPosting (example JSON-LD)

```
<script type="application/ld+json">
```

```
{
  "@context": "https://schema.org/",
  "@type": "JobPosting",
  "title": "Consultant Pediatrician",
  "description": "<p>Responsible for NICU ward...</p>",
  "identifier": {
    "@type": "PropertyValue",
    "name": "Medex",
    "value": "job_7d21e1c4"
  },
  "datePosted": "2025-10-12",
  "validThrough": "2025-11-12",
  "employmentType": "FULL_TIME",
  "hiringOrganization": {
    "@type": "Organization",
    "name": "SMS Hospital Jaipur",
    "sameAs": "https://smsjaipur.example.com"
  },
  "jobLocation": {
    "@type": "Place",
    "address": {
      "@type": "PostalAddress",
      "addressLocality": "Jaipur",
      "addressCountry": "IN"
    }
  }
}
</script>
```

Canonical & Pagination

Use `<link rel="canonical" href="...">` on paginated lists.

Use `rel="prev"`/`rel="next"` for paginated job lists.

Content Strategy

Publish keyword-focused landing pages: `"Pediatrician jobs in Jaipur"`, `"Nursing jobs by state"`.

Authority pages: statewise govt job lists, medical career guides, salary benchmarks.

FAQs & schema-rich content for featured snippets.

Use internal linking: “See similar jobs” -> improves crawl depth.

URLs & Slugs

Clean, human-readable: `/jobs/jaipur/pediatrician-consultant-sms-hospital-job_7d21e1c4`

Prefer hyphens; avoid query-heavy canonical URLs for crawled pages.

Linking & Indexing

Link to authoritative government sources on govt job pages.

Use `noindex` on preview/test pages, `noarchive` for sensitive pages as needed.

International & Multi-region

If expanding, host regionally or use CDN + hreflang. Include language selector and localized metadata.

G — Performance & Core Web Vitals Targets

Targets

LCP ≤ 2.5s (mobile).

FID / INP: aim ≤ 100ms interactive latency (minimize main-thread work).

CLS < 0.1 (avoid layout shifts).

TTFB < 600ms for SSR pages.

Strategies

Critical CSS inlined; defer non-critical styles.

Image optimization: responsive `srcset`, WebP next-gen formats, lazy-loading.

CDN for static assets; edge caching for job pages (use short TTL + webhook purge on updates).

Reduce JS bundle size: code-splitting, tree-shaking, use modern syntax and polyfills only when needed.

Use service worker for PWA and offline resume drafts.

Testing

Run Lighthouse in CI; block PR merges if Core Web Vitals regress beyond threshold.

Real User Monitoring (RUM) for LCP, FID, CLS (via web-vitals + analytics).

H — Progressive Web App (optional but recommended)

Configure manifest and service worker to allow:

Offline save of resume drafts and saved jobs.

Push notifications for job alerts.

Ensure installability and smooth offline fallback pages.

I — Content & Editorial Guidelines (for SEO + UX)

Tone: professional, helpful, and concise. Avoid sensationalist language.

Job descriptions: structured sections (Overview, Responsibilities, Qualifications, Benefits, How to Apply).

Use schema, bullets, and short paragraphs for scanability.

Govt job pages: include official reference numbers, application links, deadlines, and PDF attachments.

J — QA & Acceptance Criteria

Job Detail Page

Renders SSR HTML; includes `JobPosting` schema; LCP ≤ 2.5s; keyboard accessible; passes axe-core audit.

Search Results

Filter state reflected in URL; results update without full page reload; keyboard accessible; no content shift on filter open.

Apply Flow

Form accessible, errors announced, file upload progress, resume parse populates fields, apply success message announced.

K — Example Next.js recommendations (implementation notes)

Use `getServerSideProps` for dynamic govt job pages that must be fresh; use `getStaticProps` + `revalidate` (ISR) for most job detail pages (revalidate on job updates).

Add `<Head>` meta tags per page; build JSON-LD server-side to include in initial HTML.

Store filter state in query string; use `router.replace()` for shallow updates to avoid full reloads.

Use `next/image` with AVIF/WebP and `sizes` for responsive images.

L — SEO Monitoring & Reporting

Track:

Impressions & clicks (Search Console / GA4).

Organic job page CTR and keywords.

Crawl errors & sitemaps (Search Console).

Rich results (are JobPosting pages eligible & valid).

Index coverage for govt pages after each publish.

Automate daily checks and weekly SEO health reports.

Closing notes

Deliverables you can ask me to produce from this section:

A **component accessibility checklist** (per component) in tabular format.

JSON-LD templates for all public content types (jobs, employer, news, breadcrumb).

A **Next.js SEO starter file** (Head + JobPosting injection + ISR example).

WCAG test script (axe + Puppeteer) for CI.

16. Performance, Scalability & Reliability

This section turns non-functional requirements into concrete architecture patterns, runbooks, SLOs, operational controls and testing plans so **Medex** can reliably serve millions of users, survive traffic spikes (govt job releases), and meet business SLAs for latency and uptime.

---

1 — High-level SLOs / SLAs

Set these early and track them with SLIs & alerts.

Availability (platform): **99.9%** monthly uptime (≈43.8m downtime/year).

Search latency: median < **200 ms**; p95 < **700 ms**.

Public job page TTFB: median < **600 ms**.

API p95 (core endpoints: auth, apply, post job): < **500 ms**.

Payment success visibility: webhook processed and subscription active within **30s** of gateway event.

Background jobs: critical jobs (indexing, payments) processed within **60s** (typical), non-critical within **5 min**.

Recovery time objectives (RTO): critical service restore ≤ **2 hours**.

Recovery point objectives (RPO): ≤ **15 minutes** for transactional DB.

Adjust targets by market / SLA commitments.

---

2 — Capacity planning & sizing primer

2.1 Traffic inputs & estimates (example)

Define expected concurrent users and QPS for planning. Example formula:

`DAU` = 100,000

Peak concurrency (web) ≈ `DAU * peak_factor` (use 5%–10%) → 5,000–10,000 concurrent users.

Average page views per user/day = 6 → total PV/day = 600k

Average requests per second (QPS) = `PV/day / 86400` = ~7 QPS baseline; peak factor 10 → **70 QPS**.

For safety, plan capacity for 3× expected peak and special events (govt job publish) at 10–20×.

2.2 Sizing approach

Start with prototypes and load test (k6) to measure real resource usage per request type.

CPU & memory per pod derived from benchmarks; set HPA thresholds based on observed request CPU.

Use per-endpoint performance budgets (search heavier than profile update).

---

3 — Scalability patterns (application & infra)

3.1 Stateless services (scale horizontally)

Keep web/API servers stateless (Next.js serverless or Node/NestJS with no in-memory sessions).

Use container orchestration (Kubernetes) to scale replicas per CPU, memory or custom metric.

3.2 Stateful components (scale carefully)

Database (Postgres): single writer, read replicas for read scale. Use connection pooling (PgBouncer) to avoid connection storms. Partition large tables (jobs, applications) by date.

Search (Elasticsearch/OpenSearch): hot/warm architecture; scale by shards & replicas. Use dedicated master/data/coordinating nodes.

Redis: cache, rate-limit counters, short-lived session store; use clustering for scale and HA.

Message queue: Kafka for high-throughput durable events; RabbitMQ or SQS for simpler queue topologies.

3.3 Asynchronousization

Push heavy work to background workers (resume parsing, index updates, emailing).

Use queues with topic separation (high-priority webhooks / notifications / low-priority parsing).

---

4 — Reliability & resilience patterns

4.1 Fault isolation (bulkheads)

Isolate critical subsystems: payments, auth, search. Prevent failures in one area taking down others.

Use separate worker pools for CPU-bound (resume parse) vs I/O-bound (email) jobs.

4.2 Circuit breakers & retries

Client → API: retries with exponential backoff for idempotent operations (apply, search = no retries; payment checkout = careful).

Service → Service: use circuit breaker (Resilience4J / Polly) to avoid cascading failures.

Define retry policy per operation and ensure idempotency keys for safe retries.

4.3 Graceful degradation

If search cluster degraded: serve cached search results and display “stale results” banner.

If DB read replica lagging: route to other replica or degrade to cached API.

Payments: allow read-only billing dashboard if payment provider down; queue cache subscription changes for reconciliation.

4.4 Timeouts & resource limits

Strict request timeouts (e.g., 5s for user-facing APIs, 30s for heavier ops).

K8s resource limits + requests to avoid noisy-neighbor issues.

---

5 — Caching strategy

Edge CDN (Cloudflare / AWS CloudFront): static assets, public job pages, govt-job pages (short TTL + purge on update).

SSR / ISR caching: for Next.js, use ISR for job pages with `revalidate` and on-update purge.

Application layer cache: Redis for session store, rate-limits, frequently read fragments (employer badges, job metadata).

Query result cache: cache expensive search aggregations for short intervals (30s–5min).

Client-side caching & Service Worker: store saved searches and resume drafts offline.

Purge patterns: on job update/publish, call CDN purge + invalidate relevant cache keys.

---

6 — Database scaling & operational patterns

6.1 RDBMS (Postgres)

Primary (write) with **read replicas** (2+) to handle read traffic.

Connection handling: PgBouncer in transaction pooling mode; limit total connections.

Partitioning: monthly partitions for `jobs`, `applications`.

Backups: base backup + WAL archiving; test restores regularly.

Maintenance: autovacuum tuning, index maintenance (REINDEX when required), pg_repack for bloat.

6.2 Sharding & advanced

Only after vertical scale exhausted. Shard by region or employer_id if dataset huge.

---

7 — Search cluster design

Hot-Warm architecture: hot nodes (SSD) for recent, frequently accessed data; warm nodes for older jobs.

Replica factor: aim for at least 1 replica for high availability.

Sharding: size shards so shard size ≈ 20–50GB each; scale by adding shards/nodes.

Indexing pipeline: bulk vs single-document index; use bulk for mass imports (govt uploads), streaming for individual posts.

Snapshot & restore: daily snapshots to object storage for DR.

---

8 — Worker & queue topology

Queues:

`high-priority` (webhooks, payments) — small worker pool, immediate retries.

`default` (indexing, notifications) — medium pool.

`low-priority` (resume parsing) — large, autoscaled pool for CPU.

DLQ (dead-letter queue) and monitoring for backlog growth.

Use BullMQ (Redis) or RabbitMQ/Broker depending on needs; Kafka if ordering/retention matters.

---

9 — Autoscaling & policies

Kubernetes HPA examples

Scale on CPU plus custom queue length metric (via Prometheus Adapter).

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: api-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: medex-api
  minReplicas: 3
```

```
  maxReplicas: 25
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 60
  - type: Pods
    pods:
      metric:
        name: queue_length
      target:
        type: AverageValue
        averageValue: "50"
```

Policies:

Scale-up aggressively; **scale-down** conservatively with cooldown (5–10 minutes) to avoid thrash.

Autoscale workers by queue backlog length.

---

10 — Load testing & performance verification

Tools

k6 (recommended), Artillery, Gatling, JMeter

Scenarios to test

Baseline — steady state QPS per production estimate.

Peak — 3×–5× expected peak.

Govt publish spike — burst simulation: 10–20× baseline sustained for 30m.

Payment storm — many HRs checking out simultaneously.

Background job storm — resume parsing plus indexing backlog.

Metrics to capture

QPS, latency (p50/p95/p99), error rate, DB CPU, replica lag, queue length, GC pauses, heap usage, open connections.

Acceptance test

No 5xx errors at target load; latency SLOs met; DB replica lag < 5s at peak.

---

11 — Observability & alerting

Telemetry stack

Metrics: Prometheus → Grafana dashboards

Tracing: OpenTelemetry + Jaeger (trace requests across services)

Logs: ELK (Elasticsearch/Logstash/Kibana) or managed alternative (CloudWatch/Datadog)

Alerts: PagerDuty / OpsGenie integrated for critical incidents

Key metrics & alert thresholds (examples)

API error rate > **1%** over 5m → P1

Search p95 latency > **700 ms** → P1

Payment webhook backlog > **10** → P1

DB replica lag > **30s** → P1

Redis memory > **80%** → P2

Node CPU > **85%** → P2

Include runbook link in every alert with immediate steps.

---

12 — Incident response & runbooks

For each P1/P2, have a short runbook accessible to on-call:

Example: Search p95 high

Check ES cluster health (`/_cluster/health`).

Check indexing queue lag and recent bulk imports.

If disk pressure: add hot node or increase IOPS; scale coordinating nodes.

If spike, enable cached fallback, throttle low-priority indexing.

Post-mortem: record root cause, remediation, and latency impact.

Store runbooks in runbook repo (Confluence/Git).

---

13 — Disaster recovery & backups

Backups: DB base + WAL archived to object storage. Retain backups per policy (30 days hot, 1 year cold).

Snapshots: ES snapshots to S3 daily.

DR region: asynchronous replication to secondary region; failover documented.

DR test: quarterly restore test covering DB restore + app deploy + DNS switch.

DR plan should include DNS TTL reduction and traffic cutover commands.

---

14 — CI/CD and safe deploys

Use CI pipelines: build → test → canary deploy → gradual rollout.

Canary / Blue-Green: route small % to new revision; monitor health metrics & rollback automatically on anomalies.

Feature flags for risky features (LaunchDarkly / Unleash).

Schema migrations: zero-downtime patterns — add columns backwards-compatible, migrate data in batches, then switch reads and drop old columns.

---

15 — Cost optimization guidance

Cache aggressively to reduce search & DB costs.

Prefer managed services for ops-simplicity but monitor expensive components (ES nodes, DB I/O).

Use reserved instances / savings plans after steady baseline established.

Right-size worker pools and use spot instances for non-critical batch processing.

---

16 — Security & throttling for reliability

Rate limiting: per-user and per-IP limits (e.g., search 10 QPS/user, 100 QPS/IP).

Backpressure: return 429 with `Retry-After` for bursts; degrade to cached responses.

Auth throttles: lockout after failed attempts.

Circuit breakers for downstream (payment, search) with alerting.

---

17 — Chaos & resilience testing

Periodically run automated chaos experiments (Gremlin / Chaos Mesh):

Kill random API pod, validate HPA recovers.

Simulate ES node failure and validate failover.

Inject DB replica lag and confirm graceful fallback.

Use chaos tests in staging and pre-prod only; document blast radius.

---

18 — Observability-driven operations & capacity review cadence

Weekly: review error spikes, queue backlogs, infra utilization.

Monthly: capacity planning meeting (traffic growth, upcoming govt jobs).

Quarterly: full disaster recovery rehearsal & load test.

---

19 — Example checklist before launch (put-to-prod checklist)

Load test scenario passed at 3× peak.

DB backups & WAL archiving configured & test restore done.

ES cluster sizing validated and snapshots configured.

CDN + cache invalidation flows tested.

Payment gateway in live mode + webhook verified.

Alerts & runbooks created for top 10 incidents.

HPA configured with sensible min/max and cooldowns.

Security hardening + pentest remediation complete.

---

20 — Practical next artifacts:

k6 load test scripts for the top 5 user journeys (search, apply, HR post, checkout, gov publish).

Prometheus / Grafana dashboard templates for the SLOs above.

Kubernetes deployment + HPA + PodDisruptionBudget YAML starter files.

Runbook templates for the top 6 P1 incidents.

Example capacity planning spreadsheet with formulas tuned to your user estimates.

17. CI/CD, Staging & Deployment Strategy

This section defines a production-ready continuous integration / continuous delivery strategy for **Medex** that minimizes risk, enforces compliance, and enables frequent safe releases. It covers branching, pipelines, environments, artifacts, infrastructure-as-code, database migrations, canaries/rollouts, security scanning, rollback, runbooks and operational controls — everything your engineering + ops teams need to run reliable releases.

Summary (one-liner)

Use **pipeline-as-code** (GitOps where possible), immutable artifacts, environment parity (dev → staging → prod), gated promotion, automated testing + security scanning, and progressive deploys (canary/blue-green) backed by health checks, feature flags and clear rollback/runbooks.

---

1. Branching, versioning & release model

Branching

main (or master): production-ready, always deployable.

develop (optional): integration branch for next release.

Feature branches: feature/<ticket>-short-desc.

Hotfix branches: hotfix/<ticket> branched from main.

Merging

Pull Requests required with at least 1 code review + passing CI checks.

Require signed commits for production-affecting merges (GPG or commit signing).

Versioning

Semantic Versioning for release tags: vMAJOR.MINOR.PATCH.

Also use CI build metadata: v1.9.0+build.20251014.shaabcd1234.

Release promotion

Build once — promote artifact across envs (do not rebuild per env).

Tag artifact on successful build: medex-api:v1.9.0-build.20251014.

---

2. Environments & parity

Environments

dev — developer feature validation (ephemeral per-PR or shared).

qa / int — automated integration & contract tests; long-running testbed.

staging — production-like data/infra (same k8s/node sizes, replica counts scaled down). Used for release candidate verification.

prod — live environment, strict access controls.

Parity

Infrastructure-as-code ensures parity (same modules, different variables).

Use same container images, same versions of services, same schema migrations, but scaled to environment size.

Keep environment-specific feature flags and secrets separate.

Data in non-prod

Use **synthetic** or **scrubbed** production-like datasets. Never use raw production PII in staging without legal/contractual controls and encryption.

If a production database snapshot is needed, run an automated **sanitization** job (redact emails/phones, hash IDs) before load.

---

3. Pipeline stages (CI / CD flow)

High-level pipeline (run-as-code; example CI provider: GitHub Actions / GitLab CI / Jenkins / CircleCI)

Pre-build checks

Lint (ESLint, stylelint), format checks, commit message rules.

Dependency freeze verification.

Build & unit tests

Build Docker images for each service (API, worker, frontend).

Run unit tests.

Generate build artifact (container image + checksum).

Upload artifact to registry (ECR/GCR/Harbor), tag with semver + sha.

Static security scans (SAST + dependency)

Snyk/Dependabot, Trivy or Clair for container scanning.

Fail on critical vulnerabilities (policy driven).

Integration & contract tests (CI)

Start ephemeral test infra (docker-compose or ephemeral k8s).

Run integration tests and contract tests (API schemas).

Run DB migration dry-run in test DB.

Publish artifacts & create release candidate

Push images to registry, create release artifact record in CI.

Deploy to staging **(automated)**

Helm/ArgoCD apply of new image tag to staging.

Run smoke tests & E2E tests (Cypress / Playwright).

Run accessibility check (axe) and Core Web Vitals snapshot.

Security & compliance checks (pre-prod gating)

DAST (dynamic scan) against staging endpoints (authenticated tests).

License scan for third-party components.

Compliance checklist: backups, env vars present, secrets rotation validated.

Manual approval (gated)

After staging green, require manual sign-off from release owner, security, and product (configured in pipeline).

Deploy to prod

Progressive rollout: canary / blue-green controlled by orchestrator (Argo Rollouts / Flagger / Spinnaker).

Run production smoke tests, monitor SLOs for defined window.

Promote or roll back automatically or manually based on metrics.

Post-release tasks

Tag release in Git.

Notify stakeholders (Slack, email).

Generate runbook entry & release notes.

Schedule automated post-deploy health checks for 24–72 hours

---

4. Artifact management & immutability

Build once. Store artifacts in an immutable registry:

Containers → ECR / GCR / Harbor.

Static assets → CDN origin bucket with versioned keys.

Use immutable tags: `v1.2.3+sha` and promote by tag (avoid `latest` for prod).

Retention and garbage collection policy (e.g., keep last 30 prod images; older archived to cold storage).

---

5. Infrastructure as Code (IaC) & GitOps

IaC tools: Terraform for cloud infra; Helm + Kustomize for Kubernetes; ArgoCD for GitOps continuous delivery.

Principles

All infra changes are PRs — reviewed and tested.

Terraform state in remote backend with locking (S3 + DynamoDB lock, or Terraform Cloud).

Use modules for reusable components (VPC, RDS, EKS cluster, search cluster).

GitOps

Declarative k8s manifests stored in repo (`infrastructure/overlays/staging`, `infrastructure/overlays/prod`).

ArgoCD watches these and reconciles cluster state.

For emergency fast rollback, revert the Git commit and ArgoCD will revert cluster.

---

6. Deployment strategies & traffic shifting

Canary deployments (recommended)

Deploy new version to a small percentage of pods (1–5%), observe metrics (error rate, latency, business KPIs) for a monitoring window (5–30 minutes).

If metrics stable → increase to 25% → 50% → 100%.

If anomaly → automatic rollback.

Blue-Green

Provision new environment (green), perform smoke tests, switch load balancer to new environment atomically, keep old (blue) for quick rollback.

Rolling updates

For non-critical services, rolling update with maxUnavailable < 1 and maxSurge tuned to traffic.

Tools

Argo Rollouts, Flagger + Istio/Nginx, or Spinnaker for traffic shifting tied to metrics.

---

7. Database migrations & schema change strategy

Principles

Backward-compatible migrations only (expand/contract pattern): add new columns or table first, deploy code reading from both old & new representations, backfill data, switch reads, then drop old column in a later release.

Avoid long locks — use online schema change tools for big tables:

pg_repack, gh-ost, pt-online-schema-change for MySQL; for Postgres prefer pg_repack and partitioning strategies or logical replication approaches.

Migrations as code with a dedicated migration runner (Flyway, Liquibase, Prisma Migrate) embedded in release pipeline but executed with careful controls.

Migration workflow

Create migration file (idempotent SQL + down/backout steps).

Run against staging/test DB via CI and smoke tests.

Backup production DB (snapshot & WAL archive) before migration.

Run migration as a k8s Job with leader election (one runner) during low-traffic window when required.

Monitor job; if failure → run rollback plan from saved backup.

Rollbacks

Prefer forward-only safe migrations (rollback via new migration that undoes changes) because true DB rollback is often hard. Always prepare a tested rollback runbook.

---

8. Secrets, keys & credential handling

Secrets manager: HashiCorp Vault, AWS Secrets Manager, or GCP Secret Manager for runtime secrets.

DO NOT store secrets in Git or env files. Use CI secrets storage with least privilege.

Rotate secrets on schedule (90 days) and on suspicion/incident.

Access controls: fine-grained IAM roles for secret consumers, short-lived credentials (STS) for cloud APIs.

Audit: log all secret reads (who/when/what application).

---

9. Security & compliance gates in pipeline

Automate security checks in CI:

SAST: run on every PR (fail for high/critical).

Dependency scanning: block PRs if critical vulnerable dependency.

Container scanning: block if critical CVE.

DAST: scheduled or on-demand against staging with authenticated scans.

License & compliance checks: detect disallowed licenses.

Policy-as-code: OPA or Terraform Sentinel to prevent non-compliant infra changes.

Make security approvals explicit: for high-risk changes require sign-off from Security/Compliance before prod deploy.

---

10. Observability & automated verification (pre & post deploy)

Smoke tests (automated): simple end-to-end checks such as auth/login, search, apply, payment checkout sandbox, basic HR flows.

Health checks

Liveness probe: basic process alive.

Readiness probe: app ready for traffic (DB/redis/connectivity checks pass).

Startup probe: used to avoid killing slow-starting containers.

SLO verification

After canary step, run automated SLO checks using Prometheus queries:

Error rate, latency (p95), resource utilization.

If violation → automatic rollback.

Tracing & Logs

Add OpenTelemetry instrumentation and centralized trace dashboards to correlate CI deploys with runtime traces.

Tag logs/traces with `release_id` and `git_sha` to trace incidents to deploy.

---

11. Rollback & emergency procedures

Automated rollback triggers

Canary failure rules: error rate > configured threshold, p95 latency spike > configured threshold, or health check failing → automatic rollback to previous image.

Manual rollback

Revert to last known-good Git tag / image tag and redeploy.

Database rollback

If a schema change caused failures, follow migration rollback plan:

If impossible to rollback, restore DB from snapshot and apply forward fix or route traffic to blue environment with compatible DB.

Incident communications

On rollback, auto-notify owners with runbook and incident channels (Slack + PagerDuty).

---

12. Testing matrix (what to run where)

Test type	When	Where
Unit tests	On PR	CI runner
Lint & formatting	On PR	CI
Integration tests	On PR (optional) / CI	ephemeral infra
Contract tests	On PR	CI
E2E tests	On staging	Staging env, nightly/regression
Load tests	Before major release / periodic	Performance cluster
SAST/Dependency scan	On PR	CI
DAST	On staging	Staging
Accessibility tests	On staging	Staging

Test type	When	Where
Security/Compliance scans	Pre-prod	Staging / gated

---

13. Observability for deployments

Annotate Prometheus + Grafana with `release` tag for metrics.

Create per-release dashboards (deployment window): CPU, p95 latency, error rates, queue backlog, DB replica lag, ES health.

Use automated alert suppression during expected maintenance windows but ensure post-window verification.

---

14. Governance & approvals

Define **release owners** per release.

For major releases (DB migrations, infra changes): require approvals from Product, Security, and SRE (3-way signoff).

Record approvals in the CI run (audit trail): who approved, when, notes.

---

15. Operational runbooks (deliverables)

Create runbooks stored in an accessible runbook repo for:

Deploy checklist (pre/post).

How to promote artifact from staging → prod.

Rollback procedure (service & DB).

How to run DB restore from backup.

How to run smoke tests & health checks.

Post-mortem template.

---

16. Compliance & audit trails

CI must log builds, who triggered them, approvals, and artifact IDs.

Store pipeline logs with retention policy configurable (e.g., 90 days for CI logs, longer for compliance).

Tag deployed resources with `env`, `app`, `release`, and `owner` for auditability.

---

17. Cost & resource control (CI/CD specific)

Use ephemeral ephemeral runners / spot instances for heavy test jobs (load tests, container builds) to reduce cost.

Clean up ephemeral infra after test runs to avoid runaway costs.

Set quotas for prod deployment frequency if needed for cost forecasting.

---

18. Example minimal GitHub Actions pipeline (skeleton)

```
name: CI

on:
  push:
    branches: [ main, develop ]
  pull_request:

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Setup Node
        uses: actions/setup-node@v4
        with: node-version: 20
      - name: Install
        run: npm ci
      - name: Lint
        run: npm run lint
      - name: Unit Tests
        run: npm test -- --coverage
      - name: Build Docker Image
        run: |
          IMAGE=registry.example.com/medex-api:${{ github.sha }}
          docker build -t $IMAGE .
          echo $IMAGE > image-ref.txt
      - name: Push Image
        uses: docker/login-action@v3
        with:
          registry: registry.example.com
          username: ${{ secrets.REGISTRY_USER }}
          password: ${{ secrets.REGISTRY_PASS }}
      - run: |
          docker push $(cat image-ref.txt)
      - name: Upload Artifact
        uses: actions/upload-artifact@v4
        with:
          name: image-ref
          path: image-ref.txt
```

(Extend with SAST, container scan, infra plan/apply, and promotion workflows.)

---

19. Recommended toolchain

CI provider: GitHub Actions / GitLab CI / CircleCI / Jenkins

CD / GitOps: ArgoCD / Flux / Spinnaker / Harness

K8s deployment: Helm + Kustomize

Infrastructure: Terraform + Terragrunt (optional)

Secrets: Vault / AWS Secrets Manager

Image registry: ECR / GCR / Harbor

QA / E2E: Cypress / Playwright

Load testing: k6

SAST/DAST: Snyk / SonarQube / OWASP ZAP

Monitoring: Prometheus + Grafana + OpenTelemetry + Jaeger

Alerting: PagerDuty / OpsGenie

Policy as code: OPA (Gatekeeper) / Sentinel

---

20. Next stage deliverables:

Full **GitHub Actions + ArgoCD** pipeline files to deploy API + frontend to k8s (staging & prod).

Terraform skeleton for core infra (VPC, EKS, RDS, ECR) + remote state config.

Kubernetes manifests + Helm charts for services, including readiness/liveness probes and resource requests/limits.

DB migration runner Job manifest + example Flyway/Liquibase config and rollback plan.

Canary rollout example using Argo Rollouts with Prometheus-based analysis.

Complete deployment runbook (checklist + commands + troubleshooting steps).

18. Monitoring, Logging & Incident Response

This section is an operational, implementable playbook covering **what to monitor**, **how to log**, **tracing**, **alerts and runbooks**, **incident response processes**, and **post-incident practices**. It's written so SRE, DevOps, Security and Product teams can adopt it directly.

---

1 — Objectives & Principles

Detect problems quickly (MTTD ≤ 15 min).

Respond effectively (MTTR ≤ 2 hrs for P1).

Contain & Recover while protecting user data and revenue paths.

Learn & Improve via blameless post-mortems and action tracking.

Scaleably Alert (avoid noise; escalate on real issues).

Targets/SLOs to monitor in dashboards:

Platform availability ≥ 99.9% (monthly).

API error-rate p95 < 1%.

Search latency p95 < 700 ms.

Job publish index time < 3s.

Payment webhook processed < 30s.

---

2 — Observability Stack (recommended)

Metrics: Prometheus (ingest) → Grafana dashboards.

Logs: Centralized ELK (Elasticsearch/Logstash/Kibana) or managed Google/CloudWatch/Datadog. Ship structured JSON logs.

Tracing: OpenTelemetry → Jaeger (or vendor APM: Datadog/NewRelic).

Alerts & Ops: Alertmanager → PagerDuty/OpsGenie (escalation).

Uptime / Synthetics: Upptime / Grafana Synthetic Monitoring / Pingdom / Synthetics in Datadog.

Security logs / SIEM: Forward logs to SIEM (Elastic SIEM / Splunk / Sumo) for security correlates.

Incident Management: PagerDuty + Slack + Confluence runbooks.

Chaos & Load Testing: Chaos Mesh / Gremlin + k6 for periodic verification.

---

3 — What to instrument (SLIs & key metrics)

Service-level

Requests/sec, error rate (4xx/5xx), p50/p95/p99 latency, request throughput, CPU, memory, GC pause times.

System-level

Pod counts, node CPU/mem/utilization, disk I/O, disk usage, network errors, DB replication lag, DB connection pool usage.

Business / Product KPIs (must be visible to Product & Ops)

Job publishes / minute, job views, applications per minute, payment success rate, subscription activations, govt job publish spike rate.

Queue & background jobs

Queue length, consumer lag, retry rate, DLQ size per queue (payments, indexing, parsing).

Search & Index

ES cluster health, indexing lag, shard/unassigned shard count, search latency p95/p99.

Security

Auth failure spikes, unusual admin actions, new device logins, rate-limit exceeded events, fraud flags trending up.

---

4 — Logging strategy & best practices

Log format

Structured JSON logs (timestamp, service, env, level, trace_id, span_id, request_id, user_id (pseudonymized), message, meta).

Example fields: {"ts":"2025-10-14T12:01:02Z","svc":"api","env":"prod","level":"error","trace_id":"...","req_id":"...","msg":"payment webhook signature invalid","meta":{"gateway":"razorpay","event_id":"...","employer_id":"..."}}.

What to log

All errors and warnings, request start/stop, external API calls (with latency & response codes), admin actions (KYC approve/reject), payment webhooks, background job failures, security-critical events (auth attempts, token refresh failures).

PII handling

Do not log raw PII (emails/phone/license) except in controlled secure audit logs. Mask or hash identifiers in general logs (e.g., user_hash). Store full PII only in encrypted DB and audited access logs.

Retention

Hot logs (30–90 days) in ELK for troubleshooting; cold/archive (1–3 years) in object storage (S3) for compliance. Security logs/SIEM retention per legal requirements (7+ years for billing artifacts).

Correlation

Propagate X-Request-ID / trace_id through all services and include in logs for end-to-end traceability.

---

5 — Distributed Tracing & APM

Instrument critical flows with OpenTelemetry: auth, job publish, search, apply, payment processing, resume parsing.

Trace spans should include downstream calls (DB, ES, payment gateway).

Create tracing dashboards for common flows (e.g., apply flow trace: frontend → API → db → notifications).

Use sampling policies: higher sampling for errors and higher-everything sampling for production-critical flows (payments).

---

6 — Dashboards (recommended set)

Create Grafana dashboards with templating (env, service, region):

Overview (SRE/Home) — availability, error-rate, latency p95, active incidents.

API Health — per-endpoint QPS, p95/p99 latency, 4xx/5xx, top slow endpoints.

Search — ES cluster health, search latency, index lag.

Payments — checkout success rate, pending webhooks, chargebacks, refunds.

Background Workers — queue sizes, consumer lag, DLQ count.

DB — replication lag, connection usage, long-running queries, bloat.

Security — failed logins, admin actions, suspicious IPs.

Business Metrics — active HRs, new candidate profiles, job posts, applications/hour.

Include alert thresholds and direct links in dashboards to corresponding runbooks.

---

7 — Alerting policy & thresholds

Alert tiers

P0 / P1 (Critical) — Platform-down, payments failing, data loss, security breach. (PagerDuty, immediate paging + Slack incident channel)

P2 (High) — Degraded critical functionality (search severe slowdowns, indexing broken). (Email + Slack to on-call)

P3 (Medium) — Performance degradation or non-critical errors. (Slack digests)

P4 (Low) — Informational / trend alerts. (Email or low-priority Slack)

Example alerts

API error rate > 1% (5m) → P1.

Payment webhook backlog > 10 → P1.

DB replica lag > 30s → P1.

ES unassigned shards > 0 → P1.

Queue length > threshold for 5m → P2.

Disk usage > 80% → P2.

CPU > 90% for > 10m → P2.

Failed login spikes (rate > baseline*10) → P2 Security.

Alert best practices

Use multi-condition alerts (error rate + increased response time) to reduce noise.

Include playbook link and runbook in every alert message.

Alert deduplication window and grouping per incident.

Use severity escalation policy via PagerDuty (20m for P1 if not acknowledged).

---

8 — Incident Response Process (playbook)

Roles

Incident Commander (IC) — leads response, decisions and communications.

Scribe / Communicator — notes, timeline, and external comms.

Technical Leads — owners for API, DB, Search, Payments, Security.

On-call Engineers — execute runbook steps.

Legal / Privacy / PR — engaged for breaches or public incidents.

Incident lifecycle

Detection — Alert triggers and on-call acknowledges (PagerDuty).

Triage (10–15 min) — IC, scribe, and tech lead join incident channel (Slack). Determine scope: P1/P2 etc.

Containment (30–60 min) — Apply mitigation (circuit break, disable feature, scale up, revoke key).

Eradication & Recovery — Fix root cause or apply workaround and validate (smoke tests).

Communication — internal updates (every 15–30 min for P1) + external status page if user-visible.

Post-Incident — blameless post-mortem within 72 hours; action items tracked to completion.

Communications

Use a dedicated incident Slack channel: #incident-<id>.

Use status page (e.g., status.medex.health) to post public incident updates (initial, updates, resolution, post-mortem summary).

---

9 — Runbooks — top common incidents (quick actionable steps)

Runbook: Payment webhooks backlog (P1)

Check webhook queue backlog metric.

Inspect webhook worker logs and error type (signature, downstream error).

If signature mismatch: validate gateway secret and rotate if compromised.

If gateway sending duplicates: deduplicate by gateway_event_id; process idempotently.

Scale webhook worker pool (k8s HPA) or run emergency worker job to drain backlog.

If processing is failing due to DB write error, pause worker & investigate DB (connections/replica).

Notify finance & admin; update status page.

Runbook: Search slow/unavailable (P1)

Check ES cluster health (_cluster/health), heap usage, GC.

Check indexing queue — if huge, pause heavy bulk imports and scale ES hot nodes.

Temporarily enable cached search responses / serve stale results.

If node OOM: restart with careful draining or add nodes; check culprit query patterns.

Post-resolution: run reindex if corruption suspected.

Runbook: High error-rate on API (P1)

Identify top failing endpoint via Grafana/Kibana.

Check recent deploys (git_sha) and correlate (rollback if deploy triggered incident).

Inspect logs/traces for exceptions/stack traces.

If DB overloaded: throttle non-critical APIs, scale DB read replicas, reduce parallelism.

If transient downstream issue: circuit-break calls and degrade gracefully.Runbook: Data breach / suspected exfiltration (Security P1)

IC notifies Security + Legal. Initiate IRP: isolate systems, revoke compromised credentials, rotate keys, enable enhanced logging.

Capture forensic evidence (preserve logs & snapshots).

Notify regulators & affected users per legal timelines (72 hours for GDPR/DPDP-like policies).

Follow legal counsel guidance before public statements.

---

10 — Incident communications (templates)

Initial internal message (P1)

Incident INC-2025-001 detected: Payment webhook backlog > 50 (P1). IC: @alice. Triage channel: #incident-INC-2025-001. Immediate action: Investigate webhook worker logs. Updates every 15 min.

Public status page template

Title: Payment processing degraded — partial outage

Time detected: 2025-10-14 12:01 UTC

Impact: Some HR subscriptions & job posting activations delayed.

Status: Investigating

Next update: in 30 minutes

Resolution note

Short summary of root cause, mitigation, timeline, and user guidance (refunds, manual fixes) and link to post-mortem.

---

11 — Post-Incident & Blameless Post-Mortem

Deliverable within 72 hrs

Incident timeline with timestamps.

Root cause analysis (RCA) with evidence.

Action items with owners, target dates, and verification steps.

Severity assessment & customer impact metrics (e.g., number of affected HRs, delayed payments).

Lessons learned & follow-up (tests, additional monitoring, runbook changes).

Post-mortem meeting

Blameless culture; focus on systems & processes.

Publish summary to internal wiki & optionally a public digest for customers.

---

12 — On-call & Escalation Roster

Maintain a rotating on-call schedule (SRE + backup SRE).

Define primary/secondary rotations and escalation policy (15 min primary, 30 min secondary).

Ensure contact info in PagerDuty and Slack; include phone escalation for critical incidents.

On-call responsibilities

Acknowledge alerts within 5 minutes.

Follow runbooks; create incident channel and assign IC if none exists.

Record all actions in incident timeline.

---

13 — Testing & Validation

Playbook drills: quarterly simulated incidents with tabletop exercises.

Chaos testing: regularly run non-production chaos to validate runbooks (e.g., kill a search node).

Synthetic tests: monitor login, search, apply, checkout flows from multiple regions (RUM + synthetic).

Alert fire drills: periodically mute alerts and require manual verification to reduce complacency.

---

14 — Security & Compliance Integration

Ensure SIEM receives logs for incident forensics.

Maintain immutable audit logs for admin actions & access during incident.

Follow notification timelines per DPDP/GDPR/HIPAA when PII exposure suspected. Legal & PR must be looped in early.

---

15 — Operational Automation & Improvements

Automate common mitigations (e.g., scale webhook consumers when backlog detected, auto rotate circuit breaker on downstream failures).

Use runbook-executing bots (Slack + scripts) to run safe remediation steps (restart worker, drain node) with manual confirmation.

Track MTTR & MTTD metrics and reduce via automation.

---

16 — Playbook Templates (appendable)

Incident channel description & pinned messages.

A complete P1 runbook with commands, sample queries, and escalation contacts.

Post-mortem report template (markdown).

PagerDuty escalation policy JSON.

19. Testing Strategy & Acceptance Criteria

This section is a complete, practical testing playbook for Medex. It tells engineering, QA and product teams **what** to test, **how** to test it, **where** to run tests, and **how** to decide a feature is accepted. It includes testing types, CI gates, test data strategy, tooling suggestions, failure handling, and concrete acceptance-criteria examples (including Gherkin). Use it as your QA charter.

---

1. Testing goals & principles

Confidence: tests must give high confidence that features work and won't break critical user journeys (signup, post job, apply, payment).

Fast feedback: unit & integration tests run on PRs; longer E2E / security / perf run on staging.

Determinism: keep tests deterministic and isolated. Mock external systems where needed.

Traceability: every production requirement → automated tests + acceptance criteria.

Shift-left security & accessibility: include SAST/DAST and a11y in pipeline.

---

2. Testing types (what & why)

Unit tests — validate small pieces of logic (pure functions, services). Fast, run on every PR.

Tools: Jest / Vitest / Mocha.

Target: ~70–90% branch coverage for business logic modules.

Integration tests — verify interactions between modules (DB + app, API layer).

Use test DB + transactional rollbacks or Dockerized ephemeral DBs.

Tools: Jest with supertest, node-based integration suites.

Contract tests — ensure stable APIs between teams/services (employer API ↔ indexing worker).

Tools: Pact or Postman contract collections.

End-to-End (E2E) tests — validate full user journeys in a browser or headless environment (signup → profile → apply; HR job post → publish → candidate apply).

Tools: Cypress or Playwright.

Run: nightly + on successful staging deploys; selected critical scenarios run on every release candidate.

Acceptance tests (BDD) — Gherkin-style scenarios for product acceptance; map to E2E tests or automated steps.

Tools: Cucumber/Playwright or Cypress with Gherkin.

Performance tests — load, stress, soak tests (search & govt-publish spikes).

Tools: k6, Gatling.

Run: scheduled (weekly) and before major releases.

Security tests

SAST (every PR): Snyk / SonarQube.

DAST (staging): OWASP ZAP / Burp Suite authenticated scans.

Dependency scanning (PR): Dependabot, Snyk.

Pen-tests: quarterly / before major launches.

Accessibility tests

Automated accessibility checks (axe-core) in CI; manual screen-reader checks on staging.

Target WCAG 2.1 AA.

Chaos & resiliency tests

Fault-injection on staging (simulate ES node down, payment gateway delay).

Tools: Chaos Mesh / Gremlin (staging only).

Regression suites & smoke tests

Lightweight smoke tests run after each deploy (login, search, post job, apply, payment sandbox).

Manual exploratory testing

Periodic human exploratory sessions for UX, edge-cases, and localization.

---

3. CI/CD test gating & pipeline placement

On PR: lint, unit tests, SAST, dependency scan, small integration tests (fast). Block merge on failures.

On Merge to main / build artifact: run full unit + integration + contract tests; publish artifact if green.

On Staging deploy: run E2E acceptance, accessibility checks, DAST scans (non-blocking but must be fixed).

Release candidate: run performance smoke & security scans; manual sign-off required.

On Prod: periodic synthetic checks + RUM monitoring; no blocking tests in prod.

Set time budgets:

PR checks: ~5–15 minutes target.

Staging E2E: up to 30–60 minutes allowed but run in parallel and gated.

---

4. Test data & environment management

Environments: dev (per-PR), qa/int, staging (prod-like), prod.

Test data: use seeded synthetic data for staging; production-like but **sanitized** (PII redacted).

Fixtures: version-controlled fixtures and factories (e.g., factory-girl / pytest fixtures).

Reset strategy: transactional tests or snapshot/restore between test runs. For E2E, create ephemeral accounts and tear them down.

Secrets: test API keys for gateway sandbox accounts only; never use production keys in CI.

---

5. Mocking & external dependencies

Payment Gateways: always use sandbox/test mode and mock webhooks in PR-level tests. For staging, use actual sandbox webhooks to validate integration.

Email/SMS: use mailhog / test SMTP in dev; for staging, use real providers with limited sending to test accounts.

Third-party APIs (medical council): mock responses (both success and failure scenarios) in unit & integration; run periodic end-to-end tests against real endpoints if accessible.

Search & Indexing: for unit/integration, mock ES via test instance or in-memory search; staging should use real ES cluster.

---

6. Test automation & flaky test policy

Flaky tests: every test marked flaky must be triaged within 48 hours. Use quarantine suites for flaky tests.

Retries: limited retries for E2E (1 retry) but failing tests should be investigated — retries not a permanent fix.

Stability metric: track flakiness rate; goal <2% flaky tests across suites.

---

7. Test coverage & metrics

Coverage targets:

Unit: >70% lines on critical backend services.

Integration: cover critical business logic paths.

E2E: aim for 80–100% coverage on core user journeys (not every branch).

Test health dashboards: test pass rate, mean test runtime, flakiness rate, test lead time.

PR gate metric: require ≥95% pass on required tests before merge.

---

8. Security & compliance test specifics

SAST on PR: fail the build on critical vulnerabilities.

DAST run on staging: run authenticated scans against staging; block release only for critical vulnerabilities.

Pen tests: scheduled quarterly or pre-major launch; track remediation and retest.

Compliance checks: automated checks for PCI scope (no card data stored), and data-retention policy tests (scripts that validate rows older than retention period are redacted in non-prod snapshots).

---

9. Accessibility testing approach

Automated (CI): axe-core integrated into E2E: fail on serious violations.

Manual: screen-reader walkthroughs for all critical flows (signup, apply, job post), keyboard-only test, color contrast checks.

Acceptance: WCAG 2.1 AA compliance for all user-facing pages.

---

10. Performance testing & acceptance

Baseline tests: run k6 scripts on main flows (search queries, job pages, apply, payment) for expected load.

Spike tests: simulate govt-job publish spikes (10–20× baseline) and verify graceful degradation (cached responses).

Soak tests: run several hours to detect memory leaks and resource drift.

Acceptance: all critical endpoints meet SLOs (see Section 17 SLOs) at target load with acceptable error rate (<1%). If not, performance regression must be fixed.

---

11. Test ownership & responsibilities

Dev team: unit tests, coverage, integration tests for their services.

QA team: E2E automation, exploratory testing, acceptance tests, regression suites.

SRE/Platform: performance tests, chaos experiments, infra tests.

Security team: SAST/DAST, pentest coordination, code-vulnerability triage.

Product: define acceptance criteria + review test results for acceptance.

---

12. Acceptance Criteria: format & examples

Format (concise)

Use **Given / When / Then** (Gherkin) + explicit success criteria (performance, accessibility, security):

> **Given** [initial state]
> **When** [action]
> **Then** [expected result(s)]
> **Non-functional**: [latency / security / accessibility constraints]

Core acceptance examples

A. Candidate signup & verification

Gherkin:

```
Feature: Candidate Signup

Scenario: Candidate registers with email and completes profile
  Given an anonymous user on the signup page
  When they register with a valid email and password
  Then they receive a verification email
  And upon verifying, the account is active
  And they can complete profile and upload a resume
Non-functional:
  - Email delivered within 60s
  - Signup endpoint p95 latency < 500ms
  - Form accessible (WCAG AA)
```

Acceptance: API returns 201; DB has user row; verification token works; resume accepted (size limits), resume parsing job queued.

B. HR registration → KYC → payment → subscription activation

Gherkin:

```
Feature: Employer onboarding

Scenario: HR registers and activates subscription
  Given an HR submits org details and KYC documents
```

When admin approves KYC and HR completes payment (sandbox)
Then subscription status becomes active and job posting quota is applied
Non-functional:
- Payment webhook processed within 30s
- KYC docs uploaded scanned for viruses

Acceptance: employer.kyc_status=approved, subscription record with starts_at/expires_at created, quota updated.

C. Job search & filters

Gherkin:

Feature: Job Search

Scenario: Candidate searches for pediatric jobs in Jaipur
Given there are published pediatric jobs near Jaipur
When the candidate searches "pediatrician Jaipur"
Then results contain relevant jobs sorted by relevance and distance
Non-functional:
- Search latency p95 < 700ms
- Results include schema.org JobPosting in SSR page

Acceptance: results include job IDs matching query, distance computed, job detail SSR includes JSON-LD.

D. Candidate apply (masked)

Gherkin:

Feature: Apply masked

Scenario: Candidate applies with masked profile
Given candidate has profile and resume uploaded
When they apply to a job with apply_type=masked
Then application created with visibility=masked
And HR receives application notification but not contact details

Acceptance: application row exists; HR sees masked contact; candidate gets confirmation email.

E. Payment flow (HR checkout)

Gherkin:

Feature: Payment Checkout

Scenario: HR purchases 3-month plan via gateway
Given HR has selected 3-month plan
When they complete checkout (sandbox)
Then payment record is created and subscription active
Non-functional:
- Webhook idempotency respected
- Failed payments retried and HR notified

Acceptance: payments.status = succeeded, subscription active, invoice generated.

F. Admin posts Govt job scheduled publish

Gherkin:

Feature: Govt Job Publish

Scenario: Admin schedules govt job for future time
  Given admin creates govt_job with publish_at in future
  When publish_at arrives
  Then job status becomes published and CDN pre-warm executed
Non-functional:
  - Publish must occur at exact scheduled time ± 30s
  - Cache warmed to CDN edges pre-publish

Acceptance: job.status published, prewarm logs present, notification queued to subscribers.

---

13. Sample Gherkin E2E scenario (full example)

Feature: End-to-End Job Posting and Application

Scenario: HR posts job and candidate applies
  Given HR "apollo@example.com" is a verified employer with an active subscription
  And Candidate "dr.sara@example.com" has a completed profile
  When HR publishes job "Resident Pediatrician - Jaipur"
  Then job is searchable by "pediatrician jaipur"
  When candidate searches and applies
  Then application shows up in HR pipeline
  And candidate receives confirmation email
Non-functional:
  - Index latency < 3s
  - Job page LCP < 2.5s for mobile

---

14. Test reporting & release criteria

Release blocked if:

Any P0 or P1 functional test fails.

SAST reports critical vulnerabilities not triaged.

DAST finds critical security issue on staging.

Performance regression > 10% vs baseline on critical endpoints (unless approved by SRE).

Accessibility violations blocking WCAG AA on critical pages.

Required artifacts before release:

Passing CI status badge, test coverage report, perf summary, security scan reports, signed release checklist and product acceptance.

---

15. Test automation tech stack (suggested)

Unit / Integration: Jest / Vitest + supertest.

E2E: Cypress or Playwright (Playwright recommended for multi-browser).

Contract: Pact or Postman.

Load: k6.

SAST/Dependency: Snyk / SonarQube / Dependabot.

DAST: OWASP ZAP.

Accessibility: axe-core + pa11y; manual NVDA / VoiceOver checks.

CI: GitHub Actions / GitLab CI.

Test reporting: Allure / GitHub Test Reports.

---

16. Traceability & test management

Map each ticket / requirement to test cases in test management (Jira Xray / TestRail / GitHub Issues).

Automate test case execution linking to pipeline builds & releases.

Store test artifacts (screenshots, videos, logs) on failure; attach to ticket.

---

17. Regression prevention & continuous improvement

Enforce PR-based testing; block merges if tests fail.

Maintain regression suite of critical journeys (run on every green build).

Weekly triage of flaky tests and failing trends; track long-term quality metrics.

---

18. Example checklist for QA sign-off

All unit & integration tests pass on merge.

E2E critical journeys pass on staging.

Performance smoke tests within SLOs.

Accessibility (automated) shows no critical violations.

Security scans (SAST/DAST) show no critical issues.

Release notes reviewed & migration tested.

Backups & rollback plan validated.

Product owner signs acceptance.

20. Release Roadmap & Milestones

Below is a pragmatic, execution-ready release roadmap for Medex: phases, milestones, owners, acceptance criteria (go/no-go), risks & mitigations, release mechanics, and communication/checkpoint cadence. It's written so Product, Eng, QA, SRE, Security and GTM teams can pick up and run — adapt durations to your team size and priorities.

---

1) High-level timeline (example 6-month program)

This example assumes two-week sprints and teams already staffed. Adjust tempo to your organization.

Phase 0 — Discovery & Planning: 2 weeks

Phase 1 — Alpha / Core MVP: 8 weeks (4 sprints)

Phase 2 — Beta / Hardening: 6 weeks (3 sprints)

Phase 3 — Launch Preparation & Compliance: 4 weeks (2 sprints)

Phase 4 — Public Launch & Canary: 1 week + 2 weeks monitoring

Phase 5 — Post-Launch Stabilize & Growth: 8+ weeks (ongoing)

Total initial program: ~6 months. You can compress/expand per resources.

---

2) Phases & Milestones (detailed)

Phase 0 — Discovery & Planning (2 weeks)

Milestones

Product requirements finalized (all 23 sections reviewed).

Minimal viable feature list (MVP scope) agreed and prioritized.

High-level architecture, security & compliance plan approved.

Sprint plan + resourcing confirmed.

Owners: Product (PRD), Eng Lead (arch), Security (controls), Legal (compliance), PM (schedule)
Acceptance / Go-No-Go

PRD accepted; MVP backlog prioritized and sized.

Risk register created.

"Ready for dev" issues ≥ 6 sprints worth.

Phase 1 — Alpha / Core MVP (8 weeks)

Core deliveries

Candidate signup, profile, resume upload + resume parse pipeline (basic).

Job search & filters (basic search index + SSR job pages).

HR registration + basic KYC upload pipeline (manual admin review).

Admin panel: HR approval, job moderation, govt job posting.

Payments sandbox integration (plans, checkout flow, webhook handling).

Basic verification state machine and audit logs.

Dev infra: CI/CD, staging, monitoring basics.

Sprint outcomes (per 2-week sprint)

Sprint 1: Auth, users, profiles, DB core; CI scaffold.

Sprint 2: Resume upload + parsing worker; profile view.

Sprint 3: Jobs CRUD, draft → publish, indexer skeleton.

Sprint 4: HR onboarding, admin KYC UI, payment sandbox wiring.

Owners: Eng (frontend, backend), SRE (infra), QA (tests), Security (SAST)
Acceptance / Go-No-Go

All core APIs documented + smoke tests pass in staging.

Payment sandbox processes webhook idempotently.

Search index returns expected results for canned queries.

Penetration quick scan: no critical findings.

Demo to stakeholders.

Phase 2 — Beta / Hardening (6 weeks)

Core deliveries

Full payment production onboarding checklist (PCI scope minimized).

Subscription lifecycle automation (renewals, expiry, proration).

Masked apply flow + application pipeline + notifications.

Interview scheduling + calendar sync (basic).

Resume parsing improvements & parsing UI for corrections.

Fraud heuristics MVP + admin flagging queue.

End-to-end test suite + performance baseline runs.

Owners: Eng, QA, SRE, Security, Legal (privacy & retention checks)
Acceptance / Go-No-Go

E2E acceptance tests green on staging (critical journeys).

Load test: baseline traffic + 3× peak completed; SLOs met or documented mitigations.

DAST scan in staging: no critical vulns.

Legal signoff on data handling, third-party DPAs signed.

Phase 3 — Launch Prep & Compliance (4 weeks)

Core deliveries

Production infra hardened (WAF, KMS, SIEM).

Final pen-test & remediation (no critical or high outstanding).

Admin training, support runbooks, billing & refunds process.

Marketing & comms plan (announce, beta invites).

Beta user onboarding (invite list, feedback loop).

Prepare status page, support SLA docs.

Owners: Security, SRE, Legal, Support, Marketing, PM
Acceptance / Go-No-Go

All critical security findings resolved.

Backup/DR test & DB restore validated.

Support & on-call roster in place.

Beta cohort engaged and feedback loop ready.

Phase 4 — Public Launch & Canary (1 week + 2-week monitor)

Core deliveries

Canary release to 5–10% of traffic, monitor SLOs & metrics.

Gradual ramp to 100% if stable (canary procedure).

Marketing activation (press, targeted outreach to hospitals/associations).

Public status page updates & initial support triage.

Owners: Eng, SRE, Product, Marketing, Support
Acceptance / Go-No-Go

Canary SLO checks pass for 24–72 hours (error rate, latency, payments OK).

No P1 incidents requiring rollback.

Business KPIs (HR signups, job posts) within expected range.

Rollback Criteria

Payment failures > 1% sustained, search p95 > SLO by 2×, or new critical security issue → rollback to previous release and trigger post-mortem.

Phase 5 — Post-Launch Stabilize & Growth (8+ weeks)

Core deliveries

Feature backlog sprints: advanced search features, employer analytics, resume improvements, govt job mass-upload.

Trust & safety features: fraud model tuning, appeals flow, KYC automation.

Scalability improvements (indexing backlog optimizations).

Monetization enhancements (promos, corporate plans).

Internationalization & localization if applicable.

Owners: Product, Eng, Data, Growth, Security
Acceptance / KPIs

Monthly active HRs, conversion (signup→paid), applications per job, time-to-hire metrics tracked and improving.

---

3) Release mechanics & cadence

Sprint cadence: 2 weeks (planning → dev → demo → retro).

Releases: align to sprint boundaries; major releases following QA sign-off. Small hotfixes via urgent patch pipeline (fast rollback enabled).

Feature flags: every non-trivial feature behind flags for staged rollout and quick disable.

Artifact promotion: build once, promote same artifact from staging → prod.

Canary & metrics gates: automated checklists (error rate, latency, DB health) before progressive ramp.

---

4) Go/No-Go checklist (must pass all to launch)

Functional: all critical flows pass E2E & smoke tests.

Performance: core SLOs met at target load.

Security: no untriaged critical vulnerabilities; pen-test remediation complete.

Compliance: DPAs signed, retention policy implemented, DPO sign-off.

Operational readiness: backups verified, runbooks present, on-call roster scheduled.

Business readiness: pricing, invoices, marketing materials, support scripts ready.

Legal & Finance: payment gateway contracts signed, tax/invoicing template validated.

---

5) Roles & RACI (key stakeholders)

Product (PM) — owns roadmap, prioritization, acceptance criteria (R).

Engineering Lead — owns delivery & architecture (A).

SRE / Platform — infra, monitoring, canary execution (A).

Security / DPO — sign-off for security & privacy (C/R).

QA Lead — test strategy & gate decisions (R).

Support / Ops — runbooks & on-call (C).

Marketing / Growth — GTM materials & launch comms (C).

Finance — billing & pricing approvals (C).

CEO / Exec sponsor — final launch approval (A).

(R = Responsible, A = Accountable, C = Consulted, I = Informed)

---

6) KPIs & success metrics by milestone

Alpha: core API availability 99.5% staging; resume parse success > 80%; search relevance manual audit > 80% relevance score.

Beta: payment webhook processing < 30s; masked-apply privacy flow validated; fraud false-positive rate < 15% (manual).

Launch: Production uptime ≥ 99.9 (rolling 7d); payment success rate > 95%; HR paid conversion ≥ target (set by biz).

Post-launch (30/90 day): Monthly active HRs, jobs posted, applications per job, time-to-first-hire—track and compare vs business targets.

---

7) Risks, contingencies & mitigations

Risk: Payment provider / webhook instability
Mitigation: Multi-gateway strategy (primary + backup), idempotent webhook handling, manual billing fallback, finance playbook.

Risk: Search cluster unable to handle gov job spike
Mitigation: Pre-warm CDN & index; queue throttling; cache sticky results; emergency scale runbook.

Risk: High false positives from fraud system blocking legit HRs
Mitigation: Conservative initial thresholds, manual review SLA, fast appeals workflow.

Risk: Critical security vulnerability near launch
Mitigation: Freeze releases on critical; hotfix pipeline & rollback; communicate transparently to stakeholders.

Risk: Insufficient support capacity on launch
Mitigation: Ramp support + triage scripts, escalate to Engineering for P1s, auto-reply & status page.

---

8) Communication & stakeholder cadence

Weekly: Engineering+Product+SRE sync (status, risks, blockers).

Biweekly: Stakeholder demo + roadmap review.

Pre-launch (daily): Launch week standups + readiness checklist.

During incident: Dedicated incident channel + hourly updates for P1.

Post-launch: Executive summary at 24h, 72h, 1 week.

Customer comms: beta invites → private launch → public launch email + press; status page for outages.

---

9) Beta program & user feedback loop

Recruit 50–200 early HRs & hospitals for private beta.

Provide incentives (discounted plans, priority support).

Collect structured feedback (in-app NPS, surveys) and weekly feedback reviews.

Iterate quickly with short turnaround SEP (sprint) patches for high-impact issues.

---

10) Operational readiness checklist (pre-production go)

Production infra provisioned with monitoring + backups.

CI/CD artifact promotion tested.

Payment gateway live keys & webhook verified.

Security sign-offs (pen test + SAST).

Support playbooks & billing operations trained.

Legal & DPO approvals completed.

Runbook & rollback scripts validated in staging.

Marketing assets and documentation published.

---

11) Post-Launch cadence (first 90 days)

Day 0–7 (Launch Week): Intensive monitoring, daily sync, bug triage prioritization.

Week 2–4: Stabilization sprints (bugfixes, hardening).

Month 1–3: Growth sprints (feature improvements, analytics, performance tuning).

Quarterly: Review roadmap; prepare for next major milestone (e.g., region expansion).

---

12) Suggested immediate next steps (actionable)

Lock MVP scope for the first 4 sprints and assign sprint owners.

Staff an on-call rotation for launch (SRE + Eng).

Prepare Beta invite list (50–200)|start outreach.

Run one full staging DR test and one full payment sandbox end-to-end.

Prepare the Go/No-Go checklist owner and approval flow.

21. Team & Resourcing Estimates

Below is a concrete, runnable staffing plan you can use to hire, budget and schedule the people you'll need to build Medex from MVP → Beta → Launch → Scale. I give (a) role definitions and hiring priority, (b) recommended headcount by phase, (c) person-month effort & ramp, (d) hiring timeline mapped to the release roadmap, (e) contractor vs FTE guidance, (f) rough cost guidance and (g) onboarding / org notes and KPIs. I avoided exact salary numbers because those vary widely by country — instead I give FTE counts, hiring order, and sensible budget bands you can substitute with local rates.

TL;DR (one-sentence)

Start with a **compact cross-functional core (8–12 people)** for the MVP, then expand to ~18–28 across Beta and Launch to cover SRE, security, data, trust & safety, and growth; use contractors for burst capacity (resume parsing, performance engineering, pentest) and keep a small product/security leadership spine full-time.

A — Core roles:

Product Manager (1) — owns roadmap, acceptance criteria, prioritization, stakeholder coordination.

Tech Lead / Architect (1) — system design, API contracts, data model ownership, code reviews.

Backend Engineers (2–4) — Node.js/Next.js/Express/Nest services, DB schema, worker pipelines.

Frontend Engineers (2) — Next.js / React (SSR/ISR), accessibility, SEO implementation.

SRE / DevOps Engineer (1–2) — infra as code, CI/CD, monitoring, Kubernetes/operators.

QA / Test Engineer (1–2) — automation (E2E), performance test scripts, accessibility testing.

Security Engineer (part-time / 0.5 → 1 FTE) — threat modeling, SAST/DAST pipelines, pen-test coordination.

Data Engineer / Search Engineer (0.5 → 1) — indexing pipeline, ES tuning, vector embedding infra.

Trust & Safety / Fraud Analyst (0.5 → 1) — rules, admin workflows, manual reviews & appeals.

UX / Product Designer (0.5 → 1) — flows, accessibility, UI components, design system.

HR / Recruiter (0.2 → 0.5) — recruiting the team (use external recruiter as needed).

Customer Success / Support (0.5 → 1) — beta support, onboarding HR customers.

Finance / Billing (0.2) — invoicing, refunds, tax compliance (outsourced initially).

Legal / DPO (consultant → 0.25 FTE) — DPDP/GDPR advice, DPAs, privacy docs.

B — Headcount by phase (recommended)

Role	MVP (Phase 1)	Beta (Phase 2)	Launch (Phase 4)
Product Manager	1	1	1
Tech Lead / Architect	1	1	1
Backend Engineers	2	3	4
Frontend Engineers	2	2	3
SRE / DevOps	1	1	2
QA / Test	1	1–2	2
Security Engineer	0.5 (consult)	0.5	1
Data/Search Engineer	0.5	1	1–2
Trust & Safety	0.5 (shared)	1	1–2
UX / Product Design	0.5	1	1
Customer Success / Support	0.5	1	2
Finance / Billing	0.2	0.3	0.5
Legal / DPO	0.2 (consult)	0.2	0.25

MVP total (recommended core team): ~9–12 FTEs
Beta total: ~14–18 FTEs (ramp testers, search & fraud)
Launch total: ~18–28 FTEs (SRE scale, support, growth)

C — Person-month estimates & ramp assumptions

Ramp assumptions: new hire reaches ~50% productivity in month 1, ~75% in month 2, ~100% in month 3 for complex roles. Use these when calculating total delivery effort.

Example effort for MVP (6–8 weeks):

Backend: 2 engineers × 2 months × average productivity (0.75) → ~3 person-months effective each → total ≈ 6 pm effective.

Frontend: similar math → ~6 pm effective.

SRE / QA / PM / Tech Lead / Designer combined → ~8–10 pm effective.

(Use this model to convert product scope/estimated story points into hires if you want a precise capacity plan.)

D — Hiring priority & timeline (mapped to release phases)

T-minus 0–2 weeks (Discovery → Start of Sprint 1)

Hire/assign: Product Manager, Tech Lead / Architect, 1 Backend, 1 Frontend, QA (1), SRE (1 part), Designer (part).

Why: Start core infra, auth, DB schema, CI/CD & basic UI.

Sprint 2–3 (Weeks 3–8 — build core)

Hire: +1 Backend, +1 Frontend, increase QA to 1–2, Data/Search (contract or part-time).

Bring on Security consultant for threat model & SAST pipeline setup.

Sprint 4 → Beta (Weeks 9–14)

Hire: SRE second FTE, Trust & Safety hire (ops), Customer Success, Finance consultant.

Contract: Performance engineer (k6), Pentest vendor engaged.

Pre-launch (Weeks 15–20)

Hire: additional Backend/Frontend for scale (as needed), Full-time Security engineer, more support staff (2).

Build out growth & analytics roles later.

E — Contractor vs Full-time guidance

Prefer FTEs for these long-lived roles:

Product Manager, Tech Lead, SRE (core), Security lead, Trust & Safety lead, UX Designer.

Use contractors / agencies for short-term or burst needs:

Resume parsing / ML prototype (consultant or third-party API like Sovren/Rchilli).

Performance & load testing (k6 expert) before major launches.

Penetration testing (external vendor).

Legal/DPO assistance (consultant) until volume demands FTE.

Content/SEO writer and growth marketing for launch campaigns — agency contract works.

Fractional hires (0.2–0.5 FTE) are acceptable for Finance, Legal, and early Trust & Safety.

F — Rough cost & budget guidance — how to convert FTE to budget

Determine local fully-burdened monthly cost per FTE (total employer cost = salary + benefits + taxes + equipment). Example categories:

Tier-1 Indian mid-senior engineer fully-burdened: 60,000 **INR/month**

Multiply by headcount and months. Add ~20–30% overhead for contractors, recruiting, software licenses, and cloud. Add separate line items for:

Cloud & infra first-year baseline (managed ES cluster, RDS, CDN, storage): significant but variable by scale.

One-time costs: pentest, initial legal/DPAs, marketing launch.

If you prefer, tell me which geography and seniority you want (India vs US vs mixed) and I'll provide sample 12-month staffing cost estimates using reasonable salary bands.

G — Org structure / reporting (compact org chart)

```
CEO
└─ Product Manager
   ├─ UX Designer
   ├─ PMO / Data Analyst (shared)
   └─ Engineering
      ├─ Tech Lead / Architect
      │  ├─ Backend Eng (x3)
      │  ├─ Frontend Eng (x3)
      │  └─ Data/Search Eng (x1-2)
      ├─ SRE / DevOps (x2)
      └─ QA (x2)
└─ Security (reports to CTO/CISO; dotted to Product)
└─ Trust & Safety (reports to Product/COO)
└─ Customer Success (Support) & Finance (report to COO)
```

Small teams should keep reporting lines short and cross-functional pods aligned to feature areas (Onboarding Pod, Jobs/Search Pod, Payments Pod, Admin & Trust Pod).

H — Onboarding & ramp plan (first 90 days per hire)

Week 0–1: Admin onboarding, infra access, local dev env, read PRD, architecture walkthrough, codebase bootcamp.

Week 2–4: Assigned starter ticket + pair-programming with senior; CI pipeline runs and small production-safe changes.

Month 2: Ownership of a subdomain (e.g., resume pipeline) with mentor shadowing.

Month 3: independent delivery of features and on-call rotation inclusion.

Provide checklists per role (security, SRE, dev) and include a "first-merge" checklist to validate access & tooling.

I — KPIs to track for resourcing decisions

Track these weekly to decide hiring scale-up/down:

Sprint velocity (story points) and throughput.

Cycle time and PR lead time.

Test automation pass rates & deployment frequency.

Mean time to restore (MTTR) incidents and on-call load.

Customer support volume per support FTE (tickets/day).

Time-to-hire & recruiter funnel metrics.

If velocity or on-call incidents degrade beyond thresholds for 2 sprints, accelerate hiring.

J — Hiring & recruiting practical tips

Prioritize engineers with systems & distributed systems experience (search, queues, scaling).

For trust & safety, hire someone with moderation/fraud ops experience or from recruitment marketplaces.

Use trial contracts (3 months) for niche roles (search engineer) with option to convert.

Engage a technical recruiter or agency for senior hires; use coding + system-design interviews (take-home + paired session).

K — Contingency & contractor pools

Keep relationships ready with:

Cloud managed service partners (for Elasticsearch / search ops).

Resume-parsing vendors (Sovren, Rchilli) to avoid building ML from scratch.

Payment integrator partners and accountants for GST & invoicing.

Boutique SRE firms for emergency on-call support during launch.

---

22. Pricing & Monetization Strategy

This section defines a complete, practical pricing and monetization model for **Medex** tailored to a medical jobs marketplace (doctors, paramedical, AYUSH). It covers pricing products, upsell mechanics, billing behaviours, discounts/proration rules, tax & invoicing considerations, payment flows, fraud controls, go-to-market tactics, KPIs, and suggested experiments. Use this as the canonical commercial playbook that product, finance and growth can iterate on.

---

1. Revenue products (what you can charge for)

Core employer-facing products

Job posting credits / plans — recurring subscription plans that allow a number of active job posts and additional features (priority listing, analytics).

Pay-per-post (one-off) — single job listing for employers who don't want subscription.

Featured / Boosted listings — paid boosts to promote a job in search and email alerts (auction or fixed price).

Employer dashboard / candidate access — premium features (download resumes, contact candidates, shortlist tools).

Enterprise / Bulk packages — large hospitals, chains with multi-seat billing, SSO, API access.

Sponsored content / employer branding — company pages, logos, sponsored employer spotlights.

Candidate-facing monetization (optional / careful with optics)
7. **Resume review & premium CV services** — paid professional resume writing / parsing improvement.
8. **Background check / credential verification** — paid (or subsidized) verification service.
9. **Interview coaching / courses** — paid career services, test prep, CME courses (partnership).

Note: Candidate monetization must be optional and clearly value-add; avoid pay-to-apply.

Other revenue streams
10. **Recruitment services / contingency / retained hires** — match & placement fees (agency-style) for executive roles.
11. **APIs & data licensing** — anonymized market data and analytics for hospitals/partners (subject to privacy rules).
12. **Ads & sponsored newsletters** — carefully vetted pharma/education partners (comply with medical advertising laws).
13. **Partnerships / affiliate referrals** — e.g., background-check vendors, training providers — revenue share.

---

2. Pricing tiers — example structure (starter, scalable, enterprise)

A. Small / Starter (Monthly)

Price: **₹999** per month

Includes: 5 active job posts, basic visibility, email support, basic analytics.

B. Growth (Quarterly / 3 months)

Price: **₹2,499** per 3 months

Includes: 15 active jobs, featured in search rotation, priority email alerts, CSV export.

C. Pro (6 months)

Price: **₹4,499** per 6 months

Includes: 30 active jobs, top-of-list boosting credits (10), advanced analytics, resume downloads (limited).

D. Premium (1 year)

Price: **₹7,499**

Includes: 75 active jobs, unlimited boosts for X jobs per month, dedicated account success, integration support.

E. Enterprise (2/3/5 years)

Custom pricing (volume discounts, SLAs, SSO, API access, recruitment support).

Example: 2-year enterprise at **₹10,999** with 150 jobs, priority support, 10 seat logins.

F. Lifetime plan

One-time price (e.g., **₹29,999**) — *use sparingly*: pre-launch/backers only; must be operationally bounded (see lifetime caveats).

G. Pay-per-post

Single job: **₹499–₹1,499** depending on category and urgency.

H. Boosts

Boost credit per job: **₹199–₹999** per boost (based on prominence & duration).

---

3. Billing models & rules

Billing cadence & options

Monthly, Quarterly, 6 months, Annual, Multi-year (2y/3y/5y), Lifetime.

Payments via payment gateways (Razorpay, Stripe, PayPal). Support cards, netbanking/UPI where applicable.

Auto-renew toggle with explicit consent for recurring billing. Store tokenized payment method via gateway.

Proration & upgrades/downgrades

Upgrade mid-cycle: charge prorated difference for remaining period; immediate provisioning of new features/credits.

Downgrade mid-cycle: apply new plan at next renewal (no mid-cycle refund) or offer prorated credit (product decision).

Cancellation: allow cancellation but keep access until end of paid period; optionally offer refund windows (e.g., 7 days).

Trials & introductory offers

Offer 7–14 day free trial or limited-feature trial (1 job post). Collect card only if auto-convert; be transparent about auto-charge.

Trial eligibility: first-time HR accounts only.

Promotions & coupons

Support percent/flat coupons. Allow one coupon per purchase. Manage coupon expiry & usage in billing admin.

Refunds & disputes

Admin-managed refund requests. Policy example: full refund within 7 days of purchase (if unused), pro-rated refunds for longer durations assessed case-by-case. Document refund SLA (5–7 business days).

Invoicing

Auto-generated invoice with GST/tax information, downloadable PDF in dashboard. Include invoice numbering per accounting standard.

---

4. Tax & regulatory considerations (India focus + global notes)

India (GST)

Medex is GST-registered, charge GST on invoices for Indian employers. Tax rate varies by service (consult local tax counsel). Include GSTIN on invoices.

Maintain records for 7–10 years (for audits).

Accounting

Recognize revenue per accrual: subscription revenue recognized over service period; upfront lifetime payments recognized per accounting rules (deferred revenue amortization).

---

5. Subscription lifecycle mapping (operational)

Tie pricing into subscription lifecycle already in schema:

subscriptions record: plan_id, status (active, past_due, cancelled, expired), starts_at, expires_at, auto_renew, billing_cycle_anchor.

payments table: store gateway txn id, amount, invoice_url, status.

On payment success: create/extend subscription, increment employer job quota.

On past_due: restrict new postings; keep existing posts visible for grace period (configurable).

Dunning flow:

Retry payment automatically X times (e.g., 3 attempts over 7 days).

Send notifications: 7 days / 3 days / 1 day before suspension; after expiry, enter grace period; suspend after grace.

Offer manual renewal & support.

---

6. Fraud prevention in monetization

Fraud signals: high refund rate, multiple failed payments, chargebacks, inconsistent KYC.

Controls: require KYC approval for access to premium plans (or place limits on unverified HR accounts).

Chargebacks: detect via gateway webhooks, flag account, investigate. Keep auditable proof of service (job posting timestamps, invoices, email confirmations) to dispute chargebacks.

---

7. Enterprise & channel pricing

Enterprise contracting

SLA tiers: Standard, Priority, Dedicated. Price includes dedicated onboarding, SAML SSO, integration time, premium support. Use annual/ multi-year billing.

Volume discounts and seat-based pricing for multiple HR logins.

Channel & partner pricing

Discounts & revenue share for recruitment partners (placement agencies), medical associations, or universities. Example: 20–30% referral commission for first-year revenue or flat fee per lead.

White-label / On-prem

Possible for large hospital chains — priced as custom contract (installation, maintenance, data separation).

---

8. Promotions, acquisition tactics & pricing experiments

Early adopter pricing

Founders / Beta hospitals get steep discount (50%–80%) or lifetime early-bird. Limited time.

Referral program

Credit to referrer and referee (e.g., ₹1000 credit for both when referee purchases a plan).

Volume discounts

Scale discount tiers (e.g., 10% off at ₹100k annual spend, 20% at ₹250k).

Seasonal promotions

During hiring seasons or medical conference tie-ins — timed discounts on boosts and multi-year signups.

A/B testing & experiments

Test price elasticity: run randomized pricing experiments on new signups (ethical & legal compliance). Track conversion lift vs churn.

---

9. Candidate product pricing & ethics

Candidate paid services should be transparent, optional, and value-add (e.g., background checks, resume review).

Avoid gating essential job application features behind paywalls; that undermines trust.

Consider employer-pay-for-candidate-services (employer buys background checks as part of hiring).

---

10. Metrics & KPIs to track closely

Financial

MRR (Monthly Recurring Revenue), ARR

Average Revenue Per User (ARPU) — per paying employer

LTV (Customer Lifetime Value) — average revenue × gross margin × expected lifetime

CAC (Customer Acquisition Cost) — marketing + sales / new paying customers

CAC Payback Period — months to recover CAC

Churn Rate — monthly churn for paying employers

Net Revenue Retention (NRR) — upsell & churn combined

Gross Margin — revenue minus direct costs (payment fees, partner commissions)

Operational

Conversion funnel: signup → trial → paid → active (post jobs)

Time to first job post after payment

Boost purchase rate and revenue share from boosts

Refund rate & chargeback rate — target <1% ideally

Growth experiments

LTV:CAC ratio target typically >3 for healthy SaaS marketplace (adjust for marketplace dynamics).

---

11. Pricing governance & review cadence

Review pricing & promos quarterly.

Monthly dashboard for MRR, churn, ARPU.

Quarterly pricing experiments to optimize.

Annual price adjustments for inflation/service improvements with advanced notice (30–60 days).

---

12. Implementation notes & engineering requirements

Billing infrastructure

Maintain plans, subscriptions, payments, and invoices tables (see detailed DDL earlier).

Tokenize payment methods using gateway tokens; never store raw card data.

Implement idempotency keys for payments and webhook processing.

Invoice generation

Auto PDF generation, store S3 URL in payments.invoice_url, mark payments.status. Include GST/VAT fields.

Admin billing UI

Allow manual invoices, credits, refunds, prorate adjustments, and view payment history. Track audit logs for all billing actions.

Accounting export

Monthly ledger export (CSV) with invoices, taxes, refunds for finance team and auditors. Keep records for required retention (local laws).

Payment retry & dunning flow

Implement robust retry logic and business rules to avoid accidental double-charges. Send clear customer emails.

---

13. Customer success & retention levers

Onboarding: high-touch onboarding for Premium & Enterprise customers to reach time-to-value quickly (first hire).

Success metrics: measure job fill-rate, application rate, response time. Use these to upsell.

Upsell triggers: heavy job posters, repeat high-application volume HRs, employers with many unfilled jobs.

Discounts for upgrade: targeted offers (e.g., 20% off first-year upgrade).

Churn reduction: retention offers, account health score, proactive outreach before expiry.

---

14. Example revenue projection model (simple)

Assumptions (month 1)

Paying employers: 200 (20 Starter, 80 Growth, 70 Pro, 30 Premium)

Average monthly revenue per paying employer ≈ ₹3,000 → MRR ≈ ₹600k

Levers

Conversion rate on trials → increase paying base

Boosts & featured add-on revenue as % of MRR (20–30%)

Enterprise deals add large upfront ACV; treat separately

(Provide a spreadsheet model on request to plug in realistic numbers for your market.)

15. Risks & mitigations

Risk: Price sensitivity in public hospitals or smaller clinics.

Mitigation: Offer low-cost starter & pay-per-post; volume discount; government tie-ups.

Risk: Over-monetizing candidates (backlash).

Mitigation: Employer-focused monetization; candidate services optional & transparent.

Risk: Fraud / false postings leading to refunds and loss of trust.

Mitigation: Strong KYC & fraud detection, clear refund policies, escrow for high-value enterprise contracts.

Risk: Payment gateway constraints (dispute handling, chargebacks).

Mitigation: Multiple gateways, strong webhook handling, proper invoice/usage logs for dispute evidence.

---

16. Go-to-market pricing playbook (actionable steps)

Pilot pricing with 10–20 hospitals (discounted) to validate willingness-to-pay and collect feedback.

Launch public plans after pilot (Starter/Growth/Pro/Premium).

Offer limited-time launch discounts (e.g., 30% first year) and referral credits.

Run A/B tests for boost pricing & placement (fixed price vs auction).

Introduce enterprise sales with custom SLAs & onboarding.

Monitor KPIs and iterate pricing after 2–3 months of data.

---

17. Ethical & compliance notes

Ensure all paid features comply with medical advertising rules and local laws.

Don’t allow job postings that request upfront candidate payments; remove such posts and refund if necessary.

Maintain transparent billing and privacy terms; include DPO sign-off for monetized candidate data products.

---

18. Next actionable outputs I can generate now

a) **Pricing spreadsheet** (monthly/annual plans, projected MRR, churn impact, LTV/CAC calculator).

b) **Billing flows & sequence diagrams** (checkout, webhook, subscription creation, dunning).

c) **Sample invoice template** (GST/VAT-ready) and refund policy text for T&C.

d) **Experiment roadmap** for pricing A/B tests + KPIs to track.

23. Marketing, SEO & Growth Recommendations

This section is a tactical, end-to-end growth playbook for Medex: how to acquire employers (HRs) and candidates (doctors, paramedical, AYUSH), grow retention, build brand trust in the medical community, and scale revenue. It covers strategy, channel tactics, content & SEO plan, partnership playbook, paid acquisition, PR & events, conversion optimisation, measurement & experiments, and a 90-day launch plan + 12-month roadmap.

Guiding principles

Marketplace-first: acquire employers and candidates in balanced fashion to avoid one-sided liquidity problems.

Trust & safety = growth lever: verification badges, case studies and transparent policies convert HRs and candidates faster.

Content-led SEO: organic discovery is the highest ROI for job sites — invest early.

Localised & verticalized GTM: target hospitals, state health departments, teaching hospitals, nursing colleges and AYUSH networks by city/state.

Measure everything: funnels, cohort LTV, time-to-first-post, job-fill rate — optimize with experiments.

A. Target audiences & value propositions

Primary audiences

Employers (HR / Hospital Admins / Clinics / Labs) — need fast, qualified hires, simple billing, trust that applicants are verified.

VP: “Post jobs, reach verified medical candidates, reduce time-to-hire and administrative overhead.”

Candidates (Doctors, Nurses, Paramedical, AYUSH) — need relevant jobs, legit employers, privacy (masked apply) and easy apply.

VP: “Get matched to verified medical jobs (private & govt) and apply safely with one profile.”

Secondary & strategic audiences

Medical colleges, state medical councils, medical associations, staffing agencies, background-check vendors, CME providers.

B. Brand positioning & messaging

Core brand promise: “Medex — trusted medical jobs, faster hiring for hospitals, safer applications for professionals.”

Tone & voice: professional, concise, empathetic, trustworthy. Use credentials and verification badges as trust signals.

Hero copy examples

Employer hero: “Hire verified doctors & medical staff — faster. Post jobs, screen applicants, hire with confidence.”

Candidate hero: “Find medical jobs — private & government vacancies, one verified profile, apply in minutes.”

Top-of-Funnel messages

Employer: reduce time-to-hire, improve candidate quality, manage hiring centrally.

Candidate: verified employers, masked apply, centralized professional profile.

C. SEO & Content Strategy (high ROI — detailed)

1. Technical SEO basics (must-haves)

SSR/ISR for job and government pages (Next.js): fully-rendered HTML for crawlers.

Generate daily sitemap with <lastmod> for job/govt pages; submit to Search Console.

robots.txt to allow crawlers for public pages; block admin/staging.

Structured data: JobPosting, Organization, BreadcrumbList, FAQPage JSON-LD on relevant pages.

Canonicalization for duplicate job feeds and paginated lists.

Speed & Core Web Vitals: optimize images, use CDN, reduce LCP.

2. Keyword & content taxonomy

Build landing pages by specialty × city/state × job type:

e.g., “Pediatrician jobs in Jaipur”, “Nursing jobs in Mumbai”, “ICU nurse jobs Delhi”.

Govt jobs hub pages:

“Government medical jobs — State-wise” with curated lists and alerts.

Long-tail content: “how to apply for govt medical jobs in Rajasthan 2025”, salary guides, interview guides.

3. Content calendar & formats

Weekly blog posts (guides, how-tos): 2 posts/week to start.

Monthly in-depth reports (salary benchmarking, hiring trends) — gated for lead capture.

Templates & tools: resume templates for doctors, interview prep checklists.

Video content: 2–4 minute explainers for “How to create a profile”, “Mask apply explained.”

News & alerts feed for govt jobs — timely, authoritative.

4. On-page SEO elements

Title tags: include location + specialty + job type.

Meta descriptions: action-oriented CTAs and date freshness for govt posts.

H1 & H2 structure: “{Specialty} jobs in {City} — {#} openings” etc.

Internal linking: from specialty pages → job lists → job details → application flow.

5. Off-page & authority building

Link-building playbook: outreach to medical colleges, state health websites, local hospital sites, medical news portals for guest articles and job syndication.

Press mentions for product announcements, large enterprise deals, or sector reports.

Syndicate govt-jobs pages to aggregator partners and state portals where permitted.

6. Measurement

Rank for targeted keywords (monthly), organic sessions, job page CTR, impressions (Search Console), conversion from organic → signup → apply.

D. Content & editorial plan (first 6 months)

Weekly cadence

2 blog posts (one evergreen guide, one news/analysis).

1 newsletter (weekly digest): top new jobs, featured employers, career tips.

Monthly cadence

1 authority long-form report (salary or hiring trends) — gated lead magnet.

1 webinar or live AMA with hospital HR or senior doctor.

Sample content topics

“Top 10 hospitals hiring pediatricians in Rajasthan (2025)”

“Step-by-step guide to apply for Government Medical Jobs in India”

“How to write a medical CV that stands out”

“Salary benchmark: Consultant vs Resident vs Visiting Consultant”

“Masked apply — how your privacy is protected on Medex”

Distribution

Publish on site, share on LinkedIn/Twitter/Facebook, syndicate to medical Facebook groups, WhatsApp communities (with permission), hospital internal newsletters, and email to candidates and HR lists.

E. Paid acquisition & paid channels

Employer acquisition (high LTV, targetted)

LinkedIn Ads: target job titles (HR Manager, Recruitment Lead) and company size (hospitals, chains) — campaigns for leads (book demo) and direct signups (trial).

Google Search Ads: bids on “post medical jobs”, “hire doctors”, city-specialty queries.

Account-based Marketing (ABM): SDR outreach to large hospitals and chains (email + LinkedIn InMail).

Events / Conferences sponsorships: medical HR conferences.

Candidate acquisition (top & mid funnel)

Facebook/Instagram Ads: promote content and job alerts to medical student groups and professionals.

Google Jobs / Job boards syndication: ensure job posting schema and feed to Google for Jobs.

Paid partnerships: with medical portals, Telegram/WhatsApp channels, college placement cells.

Referral campaigns: incentives for candidates referring others.

Budget allocation (example early-stage)

40% Performance (LinkedIn + Google) — employer-focused.

30% Content & SEO (organic).

15% Events & partnerships (target hospitals/medical colleges).

10% Paid social (candidate acquisition).

5% Experimentation (misc A/B tests, influencer pilots).

Measurement

CAC by channel (employer vs candidate), conversion rate (lead → paid), cost per paid employer, ROAS for sponsored jobs.

F. Growth & activation tactics (funnel-focused)

1. Employer activation & time-to-value

Fast onboarding flow: post-first-job within 10 minutes.

Time-to-first-hire playbook: dedicated onboarding for Premium/Enterprise to guarantee first post live within X days.

Auto-suggest candidate matches and email suggestions in 24–48 hours.

2. Candidate activation

Resume parser: autopopulate profile and match to jobs.

Save searches & job alerts subscription at sign-up.

One-click apply and masked apply to lower friction.

3. Marketplace liquidity tactics

Seed employers in focus geographies: run targeted outreach to ensure there are 30–50 jobs per city/specialty at launch.

Candidate pools: partner with medical colleges and alumni networks to onboard graduates.

Hiring guarantee / trial credits: initial free postings or boosted slots to ensure visibility.

4. Viral & referral loops

Employer referrals: credits for each referred employer who converts (e.g., ₹5,000 credit).

Candidate referrals: job alert credits or premium features unlocked.

5. Retention & re-engagement

Re-engagement email flows for dormant employers (not posted in 60 days) with offers/credits.

Candidate drip: new job digests matching saved searches, “You were viewed” emails, interview invites reminders.

6. Product-led growth features

Employer dashboard metrics that highlight ROI (candidates viewed, interviews scheduled, hires) — use these to upsell.

Candidate “profile views” feature to encourage upgrades for visibility or verification.

G. Partnerships, channel & enterprise GTM

1. Partnerships to prioritize

Medical colleges & nursing schools — placement agreements; placement drives.

State Health Departments & District hospitals — syndication of govt jobs, alerts.

Medical associations & societies — co-branded webinars, newsletters.

Recruitment & staffing agencies — channel partnerships & revenue sharing.

Background-check & credentialing firms — integrated paid verification services.

2. Enterprise sales

SDR team: outbound to hospitals & chains with tiered demo + pilot offers.

Sales kit: ROI calculator (time-to-hire improvement), case studies, SLAs.

Pricing negotiation playbook & playbooks for SSO, SAML, API access.

3. Channel enablement

Partner portal with co-marketing assets, training docs, and referral tracking.

White-labelled feeds for large hospital clients.

H. PR, Events & Thought Leadership

1. PR

Pre-launch: targeted press outreach to health beat reporters, explain marketplace, pilot partners.

Launch PR: announce partnerships, enterprise customers, funding (if any), major features (govt jobs aggregator).

Use press like Times of India health, The Hindu health, LiveMint, industry blogs.

2. Events & Conferences

Sponsor or speak at healthcare HR conferences, medical college placement meets, and recruitment fairs.

Host Medex webinars featuring HR panels and successful hiring stories.

3. Thought leadership

Publish reports (salary reports, hiring trends) and pitch them to media & associations.

CEO / Product lead bylines on trusted healthcare HR topics.

I. Conversion rate optimisation (CRO)

1. Landing pages

Specialised landing pages per campaign (city + specialty) with strong CTAs, testimonials, social proof (verified employers), pricing snapshot and FAQ.

2. On-site trust signals

Verification badges, logos of hospitals using Medex, partner badges, client testimonials and case studies.

3. Forms & CTAs

Minimize fields for employer lead forms; have “book demo” / “start trial” CTA with calendar booking integration.

4. A/B tests

Test hero copy, CTA color/position, verification badge visibility, pricing phrasing (monthly vs annual), and social proof placements.

5. Checkout optimisation

Simplify payment experience (UPI, wallets) for Indian market; show tax & invoice details up front; offer promo codes.

J. Retention, upsell & lifecycle marketing

1. Email & CRM

Automate lifecycle emails:

Onboarding (Day 0–7): product tour, how to post first job.

Engagement (weekly): performance of job posts, candidate highlights.

Renewal reminders (30/14/7/1 days).

Upsell triggers: employer posted > X jobs or viewed > Y candidate profiles.

Use a CRM: HubSpot / Salesforce for employer pipeline + automated sequences for SDRs.

2. Customer success

High-touch onboarding for Premium and Enterprise customers with dedicated CSM.

Quarterly business reviews for enterprise accounts with hiring performance metrics.

3. Churn reduction

Early-warning health scores (job activity, login frequency) and outreach playbooks.

Exit surveys and targeted win-back promotions.

K. Measurement, analytics & dashboards

Core funnel metrics

Visitors → Signups → Employer activation (first job posted) → Paid conversions → Active employers (post jobs per month) → Hires reported (if tracked).

Candidate funnel: visitors → signups → profile complete → apply → interview.

Unit economics

CAC by channel, ARPU, LTV, churn, gross margin, payback period.

Operational metrics

Jobs posted/day, govt jobs posted, applications/job, time-to-fill, index latency, search CTR.

Dashboards & tools

Product analytics: Amplitude / Mixpanel for user funnels & cohorts.

Web analytics: Google Analytics 4 / Plausible.

Revenue metrics: Stripe/Razorpay + ChartMogul or Baremetrics for MRR/ARR.

CRM: HubSpot / Salesforce for pipeline and enterprise deals.

Experimentation

Maintain an experiment backlog and track results (statistical significance). Use feature-flag driven experiments (LaunchDarkly / Unleash).

L. 90-day launch marketing plan (play-by-play)

Pre-launch (Days −60 to −14)

Finalize brand assets, landing pages, SEO foundation, sitemap, job schema.

Acquire beta partners (10–30 hospitals), set up pilot success metrics.

PR outreach to announce pilot & build waitlist.

Seed content: 20 SEO landing pages for top cities & specialties.

Launch week (Day 0 to 7)

Public announcement (press release, blog), send press kit.

Paid LinkedIn + Google campaigns live (employer-targeted).

Email blast to beta list; onboard paid customers.

Host launch webinar & record for distribution.

Post-launch (Day 8 to 90)

Scale content production (3–4 posts/week), continue backlink outreach.

Run 1–2 employer-focused webinars per month.

Execute ABM & SDR outbound to prioritized hospital lists.

Begin performance experiments for pricing, landing pages, and boost product.

Build and publish first salary/hiring trends report by month 2 as a lead magnet.

M. 12-month growth roadmap (high-level)

Months 0–3: Product-market fit with 3–5 cities, initial SEO & pilot enterprise customers.
Months 3–6: Scale employer acquisition via LinkedIn & ABM, expand to 10–20 cities, launch paid boosts & featured jobs.
Months 6–9: Launch candidate premium services (optional), roll out background check integration, monetize boosts and enterprise contracts.
Months 9–12: Expand partnerships with government/state portals, start international pilot (if desired), add predictive matching & paid analytics products.

N. Marketplace & supply-side playbook (practical)

Seeding: Focus on 3–5 vertical geographies where you can guarantee sufficient jobs (e.g., state capital + medical college hub). Run direct outreach and offer free posting credits to seed supply.

Demand generation: run targeted candidate acquisition in same geos with email + social + college partnerships.

Balance metric: monitor “jobs per active candidate” and maintain target range (e.g., 1 job per 8–12 relevant active candidates) per city/specialty.

O. Recommended tools & integrations (marketing stack)

CMS & Blog: Contentful / Sanity / Next.js markdown CMS

SEO research: Ahrefs / SEMrush (or Moz for small budgets)

Email & Automation: SendGrid / Postmark + Mailchimp / HubSpot for campaigns

CRM: HubSpot / Salesforce

Analytics: GA4 + Amplitude/Mixpanel + Hotjar for UX feedback

Ad platforms: LinkedIn Campaign Manager, Google Ads, Facebook Ads Manager

Webinar: Zoom / Demio / Hopin

Social scheduling: Buffer / Hootsuite

P. Growth experiments examples (priority list)

Employer free-trial to paid conversion experiment

Offer 14-day trial with 1 free featured post. Measure conversion uplift vs control.

Boost pricing experiment

A/B test fixed price vs small auction for boosts; measure revenue & CTR lift.

SEO landing page templates

Generate 200 city-specialty pages via templating + local content → measure organic traffic lift.

Referral double-sided credit

Test employer referral credits vs cash commission to see effect on CAC.

Candidate flow friction reduction

Reduce fields on apply form; track completion rate uplift.

Verification badge impact

Show vs hide verification badges on job cards to measure employer conversion and candidate trust.

Q. KPI targets (example early-stage goals)

Month 3: 200 paying employers, 20k candidate profiles, 10k monthly job views.

Month 6: MRR target (set by your pricing) and CAC payback within 6–9 months for enterprise accounts.

Conversion: visitor→employer signup 2–3% (improve over time), employer free trial→paid 8–12% initial.

Job application rate: average 8–15 applications per job (vertical dependent).

Adjust targets to your market and revisit monthly.

R. Quick checklist to start executing this chapter (action items)

Build SEO landing page scaffold for top 20 cities & top 15 specialties.

Create 12-week content calendar and assign authors.

Run PR outreach and gather 10 pilot hospitals for beta.

Set up analytics + event tracking (Amplitude + GA4) and funnels.

Build employer onboarding email sequence & product tour.

Launch LinkedIn ad campaign targeting HR titles (pilot budget).

Prepare webinar + case study templates for enterprise outreach.

---

24 — Appendices

Below are ready-to-copy appendices you can drop into docs, pipelines and templates: (A) Email templates (HR, candidate, payments, moderation/appeals), (B) Sample PostgreSQL indexes & short notes, (C) Sample Elasticsearch / OpenSearch index mappings (jobs, profiles, employers) including analyzers and vector field examples, and (D) Glossary of terms used across the Medex project.

A — Email templates (copy & paste; replace {{...}} placeholders)

1. HR Onboarding — Welcome & next steps

Subject: Welcome to Medex — Next steps to post your first job

Hi {{hr_name}},

Welcome to Medex — your trusted partner for hiring medical professionals.

Quick steps to post your first job:
1. Complete your organization profile: {{dashboard_url}}/employer/{{employer_id}}/settings
2. Upload KYC documents (Business Registration, Authorized Signatory ID)
3. Choose a plan and post your job: {{dashboard_url}}/employer/{{employer_id}}/post-job

If you want a walkthrough, book a 20-min slot with our onboarding team: {{calendar_link}}.

Thanks,
The Medex Team
support@medex.health

---

2. HR KYC Approved

Subject: KYC Approved — You're verified on Medex

Hi {{hr_name}},

Good news — your organization KYC has been approved. You can now:
• Post jobs
• Access applicant resumes (for jobs you post)
• Use premium features included in your plan

Post your first job here: {{dashboard_url}}/employer/{{employer_id}}/post-job

If anything looks off, reply to this email and we'll assist.

Regards,
Medex Trust & Safety
trust@medex.health

---

3. Candidate — Email verification / welcome

Subject: Verify your email to activate your Medex profile

Hi {{candidate_name}},

Welcome to Medex — create one verified profile and apply to jobs across hospitals and clinics.

Please verify your email to activate your account:
{{verification_link}}

Need help completing your profile or uploading your resume? Visit: {{profile_url}}

Best,
Medex Team
help@medex.health

---

4. Payment receipt / invoice

Subject: Payment received — Invoice #{{invoice_number}}

Hi {{hr_name}},

Thank you for your payment. Here are the details:

Invoice #: {{invoice_number}}
Plan: {{plan_name}}
Amount: {{currency}}{{amount}}
Transaction ID: {{transaction_id}}
Date: {{payment_date}}
Expires: {{expires_at}}

Download invoice: {{invoice_url}}
If you need a GST update or a modified invoice, reply to this email.

Regards,
Medex Billing Team
billing@medex.health

---

5. Job removed / policy violation notice (to HR)

Subject: Action taken — Job "{{job_title}}" removed

Hi {{hr_name}},

We've removed the job posting "{{job_title}}" (ID: {{job_id}}) because it violated our policy:

Reason: {{reason_summary}}

Next steps:
1. Edit the job to comply: {{edit_link}}
2. If you believe this was in error, submit an appeal: {{appeal_link}}

We aim to review appeals within 72 hours.

Sincerely,
Medex Trust & Safety
trust@medex.health

---

6. Candidate — Application confirmation

Subject: Application received — {{job_title}} at {{employer_name}}

Hi {{candidate_name}},

We've received your application for "{{job_title}}" at {{employer_name}}.

Application ID: {{application_id}}
Submitted: {{submitted_at}}
Status: Submitted

What happens next:
• The employer reviews applications and will contact shortlisted candidates.
• You can track status here: {{applications_url}}

Good luck!
Medex — Job Alerts Team
alerts@medex.health

---

7. Appeal response (to user)

Subject: Appeal update — {{appeal_id}} ({{status}})

Hi {{requester_name}},

Thank you for your appeal regarding {{entity}} (ID: {{entity_id}}).
Status: {{status}}
Decision summary: {{decision_text}}

If you disagree with this decision, you may request a secondary review via: {{escalation_link}}

Regards,
Medex Support
support@medex.health

B — Sample PostgreSQL indexes & notes (copy into migrations)

Notes: tune these to your workload. Use CONCURRENTLY for production index creation to avoid locks. Replace table/column names to match your schema.

1) Full-text search (GIN on tsvector for job title + description)

```
-- add a computed tsvector column for faster indexing (optional)
ALTER TABLE jobs ADD COLUMN search_vector tsvector;

-- backfill (one-time)
UPDATE jobs
SET search_vector = to_tsvector('simple', coalesce(title,'') || ' ' || coalesce(description,''));

-- create GIN index
CREATE INDEX CONCURRENTLY idx_jobs_search_vector ON jobs USING GIN (search_vector);

-- trigger to keep search_vector updated
CREATE FUNCTION jobs_search_vector_trigger() RETURNS trigger AS $$
BEGIN
  NEW.search_vector := to_tsvector('simple', coalesce(NEW.title,'') || ' ' || coalesce(NEW.description,''));
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_jobs_search_vector
BEFORE INSERT OR UPDATE ON jobs
FOR EACH ROW EXECUTE PROCEDURE jobs_search_vector_trigger();
```

2) Geo queries — GIST on geography

```
-- assuming jobs.location is geography(Point)
CREATE INDEX CONCURRENTLY idx_jobs_location_geog ON jobs USING GIST (location);
```

3) Partial index for published jobs (speeds up published-only queries)

```
CREATE INDEX CONCURRENTLY idx_jobs_published_postedat
  ON jobs (posted_at DESC)
  WHERE status = 'published' AND posted_at IS NOT NULL;
```

4) Compound index for employer listing queries

```
CREATE INDEX CONCURRENTLY idx_jobs_employer_status_postedat
  ON jobs (employer_id, status, posted_at DESC);
```

5) GIN index for JSONB filters (e.g., specialties array in JSONB)

```
-- specialties stored as JSONB array in jobs.metadata->'specialties'
CREATE INDEX CONCURRENTLY idx_jobs_metadata_specialties
  ON jobs USING GIN ((metadata->'specialties'));
```

6) Application counts (denormalized counter index)

```
-- index for queries finding applicants by job & status
CREATE INDEX CONCURRENTLY idx_applications_job_status_created
  ON applications (job_id, status, created_at DESC);
```

7) Unique constraint and index for subscriptions (idempotency)

```
ALTER TABLE subscriptions
ADD CONSTRAINT unq_employer_plan_unique UNIQUE (employer_id, plan_id);
-- index added as part of constraint; adjust as necessary for multiple active subscriptions rules
```

8) Expression index for case-insensitive email lookup

```
CREATE INDEX CONCURRENTLY idx_users_email_lower ON users ((lower(email)));
```

9) Monitoring & maintenance index suggestions

```
-- index to find recently modified jobs (for caching warmers)
CREATE INDEX CONCURRENTLY idx_jobs_updated_at ON jobs (updated_at DESC);
```

C — Sample Elasticsearch / OpenSearch mappings & settings

> Use appropriate client tooling to create indices. Below are examples for ES 7+ compatible cluster. Tune analyzers per language needs.

1) Index settings (common analyzers + ngram for autocomplete)

```
{
 "settings": {
   "analysis": {
     "filter": {
       "autocomplete_filter": {
```

```
        "type": "edge_ngram",
        "min_gram": 2,
        "max_gram": 20
       }
      },
      "analyzer": {
       "autocomplete": {
        "type": "custom",
        "tokenizer": "standard",
        "filter": ["lowercase", "autocomplete_filter"]
       },
       "english_stemmer": {
        "type": "custom",
        "tokenizer": "standard",
        "filter": ["lowercase", "english_stop", "porter_stem"]
       }
      },
      "filter": {
       "english_stop": {
        "type": "stop",
        "stopwords": "_english_"
       },
       "porter_stem": {
        "type": "stemmer",
        "name": "porter"
       }
      }
     }
    }
}
```

---

2) jobs_index mapping (text, keyword, geo_point, dense_vector)

```
{
 "mappings": {
  "properties": {
   "job_id": { "type": "keyword" },
   "title": {
    "type": "text",
    "analyzer": "english_stemmer",
    "fields": {
     "raw": { "type": "keyword" },
     "autocomplete": { "type": "text", "analyzer": "autocomplete" }
    }
   },
   "description": { "type": "text", "analyzer": "english_stemmer" },
   "specialization": { "type": "keyword" },
   "employment_type": { "type": "keyword" },
   "salary_min": { "type": "integer" },
   "salary_max": { "type": "integer" },
   "experience_years": { "type": "integer" },
   "hospital_name": { "type": "text", "analyzer": "standard", "fields": { "raw": { "type": "keyword" } } },
   "city": { "type": "keyword" },
   "country": { "type": "keyword" },
   "location": { "type": "geo_point" },
   "posted_at": { "type": "date" },
   "is_government": { "type": "boolean" },
   "boost": { "type": "float" },
   "vector_embedding": { "type": "dense_vector", "dims": 384 }
  }
 }
}
```

Notes: use autocomplete field for search suggestions. Use boost to store employer-paid boost score.

---

3) profiles_index mapping (candidate profiles)

```
{
 "mappings": {
  "properties": {
   "profile_id": { "type": "keyword" },
   "full_name": { "type": "text", "analyzer": "standard", "fields": { "raw": { "type": "keyword" } } },
   "specialization": { "type": "keyword" },
   "skills": { "type": "text", "analyzer": "standard" },
   "experience_years": { "type": "integer" },
   "preferred_cities": { "type": "keyword" },
   "availability": { "type": "keyword" },
   "verified": { "type": "boolean" },
   "last_active_at": { "type": "date" },
   "vector_embedding": { "type": "dense_vector", "dims": 384 }
  }
 }
}
```

---

4) employers_index mapping (hospitals / orgs)

```
{
 "mappings": {
  "properties": {
   "employer_id": { "type": "keyword" },
   "name": { "type": "text", "analyzer": "standard", "fields": { "raw": { "type": "keyword" } } },
   "domain": { "type": "keyword" },
   "city": { "type": "keyword" },
   "country": { "type": "keyword" },
   "verified": { "type": "boolean" },
   "rating": { "type": "float" },
   "tags": { "type": "keyword" }
  }
 }
}
```

---

5) Example hybrid search query (keyword + vector re-rank)

```
{
 "query": {
  "bool": {
   "must": [
    {
     "multi_match": {
      "query": "pediatrician jaipur government",
      "fields": ["title^3", "description", "hospital_name"]
     }
    }
   ],
   "filter": [
    { "term": { "is_government": true } },
    { "range": { "experience_years": { "gte": 2 } } }
   ]
  }
 },
 "rescore": {
```

```
  "window_size": 50,
  "query": {
   "rescore_query": {
    "script_score": {
     "query": { "match_all": {} },
     "script": {
      "source": "cosineSimilarity(params.query_vector, 'vector_embedding') + 1.0",
      "params": {
       "query_vector": [ /* embedding array */ ]
      }
     }
    }
   },
   "query_weight": 0.7,
   "rescore_query_weight": 1.3
  }
 }
}
```

D — Glossary (concise definitions)

Applicant / Candidate — person applying for jobs (doctors, nurses, paramedical, AYUSH).

Employer / HR — organization account (hospital, clinic, lab) posting jobs.

Govt Job — government-run vacancy posted by admin (verified & authoritative).

Private Job — job posted by HR / employer (private sector).

KYC — Know Your Customer; documents to verify organization identity.

Masked Apply — candidate applies without exposing contact details until shortlisted.

Subscription / Plan — paid product granting job-posting rights & features.

Boost / Featured Listing — paid promotion raising a job's visibility.

Job Index — search engine index storing job documents for fast retrieval (Elasticsearch).

Vector Embedding — numeric representation of text (BERT/MiniLM) used for semantic search.

TSVector — Postgres full-text search vector type.

GIN / GIST — Postgres index types (GIN often for full-text & JSONB; GIST for geo).

ISR (Incremental Static Regeneration) — Next.js technique to statically generate pages and revalidate.

SLO / SLI — Service Level Objective / Indicator; metrics for reliability (latency, availability).

RPO / RTO — Recovery Point Objective / Recovery Time Objective (backup/DR metrics).

Dunning — automated retry & communication flow when subscription payment fails.

Audit Log — immutable record of actions for compliance and forensics.

Fraud Score — numeric score representing risk of fraudulent employer/activity.

RBAC — Role-Based Access Control; defines permissions per role (candidate, HR, admin).

PII — Personally Identifiable Information (email, phone, registration numbers).

DAST / SAST — Dynamic / Static Application Security Testing.

WAF — Web Application Firewall.

CDN — Content Delivery Network (edge caching for site assets).

PGBOUNCER — Postgres connection pooler.

SIEM — Security Information and Event Management (centralized security logs & alerts).

Quick copy checklist (what to drop in repo right away)

Add the email templates to transactional email service (Handlebars or similar).

Add the Postgres indexes as migrations (use CONCURRENTLY in prod).

Create ES index templates with analyzers and mappings (adjust dims for embeddings).

Add glossary to product docs & onboarding materials.
