WBS (Task List)

5 roles (Role 1–Role 5) working in parallel where possible

- Role 1: AI Integration
- Role 2: Backend Developer
- Role 3: Frontend Developer
- Role 4: Product & Demo Lead
- Role 5: Integration & Testing Lead

WBS: Work Breakdown Structure

Phase 1. Initiation, Environment, and Foundations (P1)

Goal: Establish the development environment, project structure, and a running base API scaffold to support ingestion, persistence, and basic endpoints.

Phase 1 Deliverables

- Repos and CI/CD scaffolding
- Local development DB and storage structure
- Base Spring Boot API skeleton with core controllers
- Coding standards and starter READMEs

P1-01 Kickoff and alignment

- Role: All
- Summary: Confirm sprint goals, success criteria, definition of done, key risks, and communication plan.
- Sub-tasks:
  - P1-01a Schedule and hold kickoff meeting
  - P1-01b Document sprint charter, success metrics, and DoD
  - P1-01c Identify stakeholders, owners, and handoff points
  - P1-01d Risk log creation and mitigation plan
- Deliverables: Sprint charter, risk log, communication plan
- Est: 2h
- Dependencies: None
- Acceptance criteria:
  - Charter approved by all required stakeholders
  - Clear DoD and success metrics documented
  - Risks captured with owner and mitigation actions

P1-02 Environment bootstrap

- Role: Role 2 (Backend), Role 3 (Frontend) with collaboration
- Summary: Create repositories, initialize CI/CD, set up local Postgres, create local filesystem structure for resume storage, and define naming /convention standards.
- Sub-tasks:
  - P1-02a Create/clone repos: `backend`, `frontend`, `shared-lib`
  - P1-02b Define branch strategy and PR process
  - P1-02c Add Docker Compose for local DB (Postgres) and a simple file-storage service (mock S3)
  - P1-02d Create initial project structure (modules: `api`, `domain`, `infrastructure`, `service`)
  - P1-02e Add sample seed data and seed scripts (for quick demo)
  - P1-02f Establish basic logging/monitoring stubs
- Deliverables: Repos wired, Docker Compose, initial folders, seed data
- Est: 3h
- Dependencies: P1-01
- Acceptance criteria:
  - Local DB container runs and is reachable
  - Project builds with a minimal “Hello API” endpoint
  - README with local setup steps is present

P1-03 Base API skeleton

- Role: Role 2
- Summary: Create a runnable Spring Boot project with core controllers and minimal, well-typed DTOs for the main entities (Candidate, Resume, Template, Score). Establish a shared `api` module and a simple security config scaffold.
- Sub-tasks:

    - P1-03a Define base package structure: `com.company.app.api`, `com.company.app.domain`, `com.company.app.service`, `com.company.app.infra`
    - P1-03b Implement core controllers: `ResumesController`, `CandidatesController`, `TemplatesController`, `ScoresController` with placeholder methods
    - P1-03c Create common DTOs: `CandidateDTO`, `ResumeDTO`, `TemplateDTO`, `ScoreDTO` (minimal fields)
    - P1-03d Add request/response validation annotations
    - P1-03e Add OpenAPI/Swagger configuration (optional but recommended)
    - P1-03f Create basic unit test skeletons for controllers
- Deliverables: Running API skeleton, basic controllers, DTOs
- Est: 3h
- Dependencies: P1-02
- Acceptance criteria:
    - Project builds and runs
    - Endpoints exist with proper request/response shapes
    - Basic tests compile and run

Phase 2. Ingestion, Data Model, and Persistence (P2)

Goal: Define the data model, implement resume ingestion API (async), and establish batch processing and basic auditability storage.

Phase 2 Deliverables

- Domain model (JPA entities) for core objects
- Resume ingestion API with async processing trigger
- Batch ingestion tracking and status endpoints
- ResumeDocument persistence and linkage to CandidateProfile
- AuditLog scaffolding and ingestion hooks

P2-01 Data model definitions (CandidateProfile, ResumeDocument, EducationRecord, ExperienceRecord, Skill, Certification, Publication)

- Role: Role 2
- Summary: Design and implement JPA entities and relationships; select PK strategy (UUID recommended); define enums and value objects
- Sub-tasks:
    - P2-01a Define domain boundaries and aggregate roots
    - P2-01b Create JPA entities with relationships:
        - `CandidateProfile` 1-to-many with `EducationRecord`, `ExperienceRecord`, `Skill`, `Certification`, `Publication`
        - `ResumeDocument` 1-to-1 or 1-to-many with `CandidateProfile` (as appropriate)
    - P2-01c Define PK strategy (UUID), auditing fields (createdAt, updatedAt)
    - P2-01d Create enums: `ParsingStatus`, `ProficiencyLevel`, `Role` (if used), etc.
    - P2-01e Define indexes for query performance: candidateId, resumeId, templateId
    - P2-01f Create Liquibase/Flyway migration scripts (baseline schema)
- Deliverables: Entity classes, migration scripts, and a simple in-memory repository mock for tests
- Est: 3h
- Dependencies: P1-03
- Acceptance criteria:
    - Entities reflect relationships and constraints
    - Schema can be created and seeded in a local DB
    - Basic unit tests for entity mappings exist

P2-02 Resume ingestion API (POST /api/v1/resumes/upload)

- Role: Role 2
- Summary: Accept resumes in batch or single payload; trigger asynchronous ingestion pipeline
- Sub-tasks:
    - P2-02a Define API contract (multipart and/or base64 JSON option)
    - P2-02b Implement `ResumesUploadController` with endpoint:
        - POST /api/v1/resumes/upload
    - P2-02c Implement `ResumeIngestionService` with async processing hook (simulated queue or `@Async`)
    - P2-02d Validate input types and sizes, reject unsupported formats
    - P2-02e Persist batch metadata and initial batch state
    - P2-02f Emit an audit log for ingestion start
- Deliverables: Endpoint, async pipeline trigger, batch metadata creation
- Est: 4h
- Dependencies: P2-01
- Acceptance criteria:
    - Endpoint accepts input and returns a batch_id with status
    - Ingestion is kicked off asynchronously
    - Audit log entry created for ingestion start

P2-03 Batch processing scaffolding

- Role: Role 2
- Summary: Implement batch status tracking and simple progress reporting
- Sub-tasks:
  - P2-03a Define `Batch` entity/model: `batchId`, `submittedAt`, `total`, `completed`, `failed`, `status`
  - P2-03b Create in-memory or DB-based progress tracker
  - P2-03c Implement `GET /api/v1/resumes/batch/{batch_id}/status`
  - P2-03d Implement retry logic for failed parses (mark as retriable)
  - P2-03e Integrate batch status with ingestion flow and audit logs
- Deliverables: Batch tracking API, progress updates
- Est: 3h
- Dependencies: P2-02
- Acceptance criteria:
  - Batch status endpoint returns accurate totals and progress
  - Failed items are retried automatically (or flagged for retry)
  - Progress is observable from UI or API calls

P2-04 ResumeDocument linkage and persistence

- Role: Role 2
- Summary: Persist raw resume metadata and link to the candidate profile; store file metadata
- Sub-tasks:
  - P2-04a Define `ResumeDocument` fields: `resumeId`, `candidateProfile` FK, `fileName`, `fileType`, `size`, `ingestionTimestamp`, `parsingStatus`, `extractSummary`, `storageLocation`
  - P2-04b Map `ResumeDocument` to `CandidateProfile` (nullable until candidate is created)
  - P2-04c Create repository, CRUD, and basic tests
  - P2-04d Ensure file storage location is captured (local path for sprint)
- Deliverables: `ResumeDocument` entity, repository, persistence logic
- Est: 2h
- Dependencies: P2-01
- Acceptance criteria:
  - ResumeDocument rows persist with correct FK to CandidateProfile (or null if not yet created)
  - Metadata fields (name, type, size, location) stored and retrievable

P2-05 Auditability hooks (AuditLog)

- Role: Role 5
- Summary: Introduce an audit trail for ingestion and early data actions
- Sub-tasks:
  - P2-05a Define `AuditLog` entity: `logId`, `userId` (or system), `action`, `targetType`, `targetId`, `timestamp`, `details`
  - P2-05b Create audit logging service/utilities
  - P2-05c Instrument key ingestion steps to emit audit records
  - P2-05d Provide ad-hoc audit query helpers or views
- Deliverables: `AuditLog` table, logging hooks, sample queries
- Est: 2h
- Dependencies: P2-02
- Acceptance criteria:
  - Ingestion actions produce audit records with meaningful context
  - Audit data can be queried for a given batch or candidate

Phase 3: Resume parsing and AI integration

Goal: Implement resume parsing via Apache Tika, shape AI outputs into domain models, attach per-field confidence, and establish parsing auditability. This phase wires the core parsing flow into the CandidateProfile domain.

P3-01: Apache Tika integration for resume text extraction

- Roles: Role 3 (Frontend) primary, Role 2 (Backend) support
- Summary: Set up a robust text extraction pipeline to pull raw text from PDFs/DOCs/TXTs and expose a minimal API for downstream parsing.
- Sub-tasks:
  - P3-01a Prepare Tika container/service (config, dependencies)
  - P3-01b Implement ResumeTextExtractionService (invokes Tika, returns plain text)
  - P3-01c Expose extraction endpoint or hook into ingestion flow
  - P3-01d Add error handling and fallback paths for unreadable files
- Deliverables: Tika-based parser service, test files, basic API
- Deliverables: Text extraction results stored or passed downstream
- Est: 3h
- Dependencies: P2-01
- Acceptance criteria:
  - PDF/DOP/DOCX/TXT inputs yield non-empty extracted text (or meaningful error)
  - Extraction results are retrievable by downstream stages
  - Basic logs generated for success/failure cases
- Owner: Role 3 (Frontend) with Role 2 support

P3-02 AI parsing prompts and data shaping

- Roles: Role 1 (AI Integration) primary
- Summary: Create centralized AI prompts to extract candidate fields and map outputs to domain objects.
- Sub-tasks:
  - P3-02a Define output schema for parsed candidate data (e.g., CandidateProfile fields with nested Education/Experience/Skills)
  - P3-02b Develop AI prompts for parsing (field-level extraction, confidence hints)
  - P3-02c Implement data-mapping layer: AI outputs -> CandidateProfile, EducationRecord, ExperienceRecord, Skill, Certification, Publication
  - P3-02d Validation and error handling for AI outputs (schema validation, fallback to human review if major gaps)
- Deliverables: Prompt templates, output schema, mapping logic
- Est: 4h
- Dependencies: P3-01
- Acceptance criteria:
  - AI output conforms to defined schema or triggers validation errors
  - Mapped domain objects populate consistently
  - Confidence data is attached per field for downstream use
- Owner: Role 1

P3-03 Field-level confidence scoring

- Roles: Role 1 (AI Integration)
- Summary: Attach per-field confidence scores to parsed data and surface low-confidence fields to UI/human review.
- Sub-tasks:
  - P3-03a Define per-field confidence model (e.g., name, email, education, skills)
  - P3-03b Extend CandidateProfile schema with confidence map
  - P3-03c Persist confidence data alongside parsed fields
  - P3-03d Provide hooks for UI to highlight low-confidence fields
- Deliverables: Confidence map structure, persistence, UI hooks
- Est: 2h
- Dependencies: P3-02
- Acceptance criteria:
  - Confidence scores exist for key fields
  - UI can highlight low-confidence fields and prompt human review
- Owner: Role 1

P3-04 Parsing audit logs

- Roles: Role 5 (Audit & Compliance)
- Summary: Add an auditable trail for resume parsing events.
- Sub-tasks:
  - P3-04a Define audit events: parsing_started, parsing_completed, per-field_confidence, errors
  - P3-04b Persist AuditLog entries for parsing actions
  - P3-04c Provide quick-lookup queries for audit history
- Deliverables: AuditLog table/schema, logging utilities, sample queries
- Est: 1h
- Dependencies: P3-03
- Acceptance criteria:
  - Parsing actions are auditable with timestamps, identifiers, and context
- Owner: Role 5

Phase 4: Job templates, templates API, and AI-assisted creation

Goal: Define and version job templates, enable AI-assisted creation of templates, and expose template APIs.

P4-01 JobTemplate data model (templateId, version, title, skills, constraints, etc.)

- Roles: Role 2 (Backend) primary
- Summary: Create the core JobTemplate entity with versioning and basic fields.
- Sub-tasks:
  - P4-01a Define domain model for JobTemplate (templateId UUID, version int, title, seniorityLevel, requiredSkills, experienceBands, educationRequirements, locationPreferences, languages, certifications, domainConstraints, createdAt, updatedAt, createdBy)
  - P4-01b Define relationships to related entities as needed (e.g., skills as embedded vs. join table)
  - P4-01c Create DB schema and JPA entities
  - P4-01d Add simple validations (non-empty title, non-empty skills)
- Deliverables: JobTemplate entity, schema, repository
- Est: 2h
- Dependencies: P2-01
- Acceptance criteria:
  - Entity maps correctly to DB with appropriate constraints
  - Basic CRUD via API can be demonstrated

- Owner: Role 2

P4-02 AI-assisted template creation prompts

- Roles: Role 1 (AI Integration)
- Summary: Provide prompts that generate templates from natural language inputs.
- Sub-tasks:
  - P4-02a Define prompts for template creation (e.g., "Create template for Senior Java Developer")
  - P4-02b Store prompt templates in versioned repository
  - P4-02c Validate and sanitize AI outputs into the JobTemplate schema
- Deliverables: Prompts, example outputs, validation logic
- Est: 2h
- Dependencies: P3-02
- Acceptance criteria:
  - AI-produced templates adhere to defined schema
  - Outputs are idempotent and versioned
- Owner: Role 1

P4-03 Template versioning and audit

- Roles: Role 4 (Product Demo Lead) primary
- Summary: Implement versioning on templates and an audit trail for changes.
- Sub-tasks:
  - P4-03a Add versioning field and versioning rules on update
  - P4-03b Create AuditLog entries for template changes
  - P4-03c API to retrieve historical versions
- Deliverables: Versioned templates endpoints, audit trail
- Est: 2h
- Dependencies: P4-01
- Acceptance criteria:
  - Each update creates a new version
  - Historical versions retrievable and auditable
- Owner: Role 4

P4-04 Template retrieval APIs (GET by id, list, search)

- Roles: Role 2 (Backend)
- Summary: Expose endpoints to fetch templates, list all, and search by keywords/skills.
- Sub-tasks:
  - P4-04a GET /api/v1/templates/{template_id}
  - P4-04b GET /api/v1/templates
  - P4-04c GET /api/v1/templates?skills=...
  - P4-04d Basic pagination and filtering
- Deliverables: Template API endpoints, basic docs
- Est: 1h
- Dependencies: P4-01
- Acceptance criteria:
  - Endpoints return expected template data with version metadata
- Owner: Role 2

Phase 5: Intelligent Matching engine and scoring

Goal: Implement semantic matching, AI-based scoring, explainability, and the end-to-end scoring flow with human-in-the-loop.

P5-01 Semantic matching engine (resume vs template)

- Roles: Role 1 (AI Integration) primary, Role 2 (Backend) secondary
- Summary: Build a semantic matching pipeline that compares parsed resume data to a JobTemplate and yields an AI score.
- Sub-tasks:
  - P5-01a Define matching criteria and scoring schema (score components, weights)
  - P5-01b Implement semantic analysis logic (vector similarity, keyword matching, or rule-based fallbacks)
  - P5-01c Produce aiScore and aiConfidence per candidate-template pair
  - P5-01d Add logging of prompts, model version, and decisions for traceability
- Deliverables: Scoreable result with aiScore, aiConfidence, rationale hooks
- Est: 4h
- Dependencies: P3-02, P4-01
- Acceptance criteria:
  - ScoreRecord stores aiScore with a transparent confidence
  - Reasoning hooks are available for explainability
- Owner: Role 1 (AI Integration)

P5-02 Scoring compute API (POST /api/v1/scores/compute)

- Roles: Role 2 (Backend) primary
- Summary: Trigger AI scoring for one or more candidates against a template; asynchronous or synchronous as appropriate for the demo.
- Sub-tasks:
  - P5-02a Implement API accepting candidate_id, template_id, and source metadata
  - P5-02b Support single and batch scoring inputs
  - P5-02c Persist ScoreRecord with aiScore and aiConfidence
  - P5-02d Return score_id and a status field (queued/processing/complete)
- Deliverables: Score compute endpoint, status tracking
- Est: 3h
- Dependencies: P5-01
- Acceptance criteria:
  - API returns a score_id quickly; batch requests processed; status transitions observable
- Owner: Role 2

P5-03 Explainable scoring and rationale

- Roles: Role 1 (AI Integration)
- Summary: Generate human-readable explanations for AI scores, including key drivers per skill/experience.
- Sub-tasks:
  - P5-03a Create rationale model: which fields drove the score and why
  - P5-03b Attach rationale to ScoreRecord; expose through API
  - P5-03c UI surface for explainability (AI Explainability Panel)
- Deliverables: Rationale, UI hook, API endpoints
- Est: 2h
- Dependencies: P5-02
- Acceptance criteria:
  - Rationale is coherent, per-field, and helps human reviewers understand the AI decision
- Owner: Role 1

P5-04 Confidence handling and routing to human review

- Roles: Role 5 (Audit/Review) with Role 4 (Product Demo Lead) involvement
- Summary: Route low-confidence scores to human reviewers; provide a queue and feedback loop.
- Sub-tasks:
  - P5-04a Define confidence thresholds and routing rules
  - P5-04b Implement routing to a human-review queue; mark items as needing review
  - P5-04c Audit trail for routing decisions and reviewer assignments
- Deliverables: Routing logic, audit entries, reviewer queue
- Est: 2h
- Dependencies: P5-01, P5-02
- Acceptance criteria:
  - Low-confidence scores appear in the review queue with rationale
  - Review actions update the ScoreRecord and final score as needed
- Owner: Role 5 (primary), Role 4 (co-owner for UI and demo)

P5-05 Score override workflow (UI hook to post override)

- Roles: Role 3 (Frontend) and Role 4 (Product Demo Lead)
- Summary: Allow override of AI score with rationale; persist and reflect in final score.
- Sub-tasks:
  - P5-05a API for submitting an override (override_value, reason, reviewer_id)
  - P5-05b UI integration: override control, validation, and submit
  - P5-05c Persist OverrideNote and recompute final score (0.75 AI + 0.25 human/intangible by default)
- Deliverables: Override API, UI controls, audit trail
- Est: 2h
- Dependencies: P5-02
- Acceptance criteria:
  - Override accepted, persisted, and reflected in finalScore
  - Audit trail updated with override context

Phase 6: Bias detection and analytics scaffolding

Goal: Implement bias detection hooks and surface flags in UI; scaffold analytics signals to support HR insights and fairness monitoring.

P6-01 Job Description Bias Check endpoint

- Roles: Role 5 (Audit/Review) primary, Role 1 (AI Integration) support
- Summary: Expose an API to analyze job descriptions for biased language and fairness issues.
- Sub-tasks:
  - P6-01a Define item payload: item_type, item_id, text
  - P6-01b Implement /api/v1/bias/check

    - P6-01c Return flags with severity and context
- Deliverables: Bias check API, sample responses
- Est: 3h
- Dependencies: P5-02 (scoring) may not block, but data available
- Acceptance criteria:
    - API returns a structured set of bias flags and remediation suggestions

P6-02 Bias Flags Retrieval

- Roles: Role 2 (Backend)
- Summary: List and retrieve bias flags (GET /bias/flags and /bias/flags/{flag_id})
- Sub-tasks:
    - P6-02a Implement BiasFlag entity/model
    - P6-02b Implement /api/v1/bias/flags and /bias/flags/{flag_id}
- Deliverables: BiasFlag endpoints and data model
- Est: 2h
- Dependencies: P6-01
- Acceptance criteria:
    - Flags retrievable with correct item_type and IDs

P6-03 Bias Visualization UI

- Roles: Role 3 (Frontend) + Role 4 (Product Lead)
- Summary: UI panel to visualize bias flags and remediation suggestions
- Sub-tasks:
    - P6-03a UI component: BiasFlagPanel
    - P6-03b Bind to /bias/flags data
    - P6-03c UI for remediation guidance
- Deliverables: Bias visualization in Candidate Review Dashboard
- Est: 3h
- Dependencies: P6-01, P6-02
- Acceptance criteria:
    - Flags and remediation appear clearly in UI with masking for PII

P6-04 Remediation Suggestions and Automation

- Roles: Role 1 (AI Integration) + Role 5 (Audit)
- Summary: Provide concrete remediation suggestions to improve fairness in job descriptions and evaluation criteria
- Sub-tasks:
    - P6-04a Define remediation templates (e.g., replace biased phrases)
    - P6-04b Attach remediation guidance to flags
    - P6-04c Optional auto-suggested template edits (non-blocking)
- Deliverables: Remediation guidance data
- Est: 2h
- Dependencies: P6-01
- Acceptance criteria:
    - Remediation guidance exists and aligns with flags

Phase 7: Analytics and exports

Goal: Build HR analytics scaffolding, dashboards, and export capabilities with privacy controls.

P7-01 Analytics Dashboard Scaffold

- Roles: Role 5 (Analytics) + Role 2 (Backend)
- Summary: Create a basic dashboard page with charts for batch timing, scoring variance, bottlenecks, and workload
- Sub-tasks:
    - P7-01a Define metrics and data model
    - P7-01b REST endpoints to fetch metrics
    - P7-01c Frontend charts placeholders
- Deliverables: Dashboard page skeleton, API surface
- Est: 3h
- Dependencies: P5-02, P7-04
- Acceptance criteria:
    - Dashboard loads with real-time-ish data and placeholders

P7-02 Data Export Endpoints

- Roles: Role 5 (Analytics) + Role 2 (Backend)
- Summary: Expose CSV/JSON exports with redaction as configured
- Sub-tasks:
    - P7-02a Implement /api/v1/analytics/export

    - P7-02b Redaction policy hook
- Deliverables: Exports endpoint and policy
- Est: 3h
- Dependencies: P7-01
- Acceptance criteria:
    - Exports generated with redaction rules applied

P7-03 Real-time/Near Real-time Metrics

- Roles: Role 5 + Role 2
- Summary: Add near-real-time metric refresh cadence
- Sub-tasks:
    - P7-03a Add caching/refresh logic
    - P7-03b Hook metrics to batch processing events
- Deliverables: Refresh workflow
- Est: 2h
- Dependencies: P7-01
- Acceptance criteria:
    - Metrics update within a short interval (e.g., 15–30 seconds)

P7-04 Privacy-first Exports (PII redaction)

- Roles: Role 5
- Summary: Ensure exported data respects privacy settings
- Sub-tasks:
    - P7-04a Define redaction rules per field
    - P7-04b Apply redaction in export pipelines
- Deliverables: Redacted exports
- Est: 2h
- Dependencies: P7-02
- Acceptance criteria:
    - PII is redacted in exports by default or per policy

Phase 8: UI development

Goal: Complete UI surfaces for candidate review, batch upload, bias visualization, analytics, and settings/help.

P8-01 Candidate Review Dashboard UI

- Roles: Role 3 (Frontend)
- Summary: Implement list and detail views showing AI scores, confidence, rationale, and parsed data
- Sub-tasks:
    - P8-01a CandidateListComponent, CandidateDetailComponent
    - P8-01b Data bindings for /api/v1/candidates and /api/v1/scores
    - P8-01c AI Explainability panel placeholder
- Deliverables: Functional candidate UI
- Est: 6h
- Dependencies: P5-02, P5-03
- Acceptance criteria:
    - UI shows AI scores, confidence, and parsed fields

P8-02 Batch Upload UI

- Roles: Role 3
- Summary: Drag-and-drop batch uploader with progress
- Sub-tasks:
    - P8-02a BatchUploadPanel component
    - P8-02b Integrate with /api/v1/resumes/upload
- Deliverables: Batch upload UI
- Est: 3h
- Dependencies: P2-02
- Acceptance criteria:
    - Uploads produce batch_id and progress

P8-03 Bias Visualization UI

- Roles: Role 3
- Summary: Bias flag visualization and remediation guidance in UI
- Sub-tasks:
    - P8-03a BiasFlagPanel integration
- Deliverables: Bias UI panel
- Est: 2h

- Dependencies: P6-03
- Acceptance criteria:
  - Flags visible in UI with remediation text

P8-04 Analytics Dashboard UI

- Roles: Role 3
- Summary: Render analytics charts from /analytics/dashboard
- Sub-tasks:
  - P8-04a Integrate charts into Analytics page
- Deliverables: Analytics UI page
- Est: 2h
- Dependencies: P7-01
- Acceptance criteria:
  - Charts render with data

P8-05 Settings and Help UI

- Roles: Role 3
- Summary: Help center, privacy settings toggle, seed/demo controls
- Deliverables: Settings UI
- Est: 1h
- Dependencies: P7-01
- Acceptance criteria:
  - Settings available in UI and reflect policy

Phase 9: Security, compliance, and audit

Goal: Harden mock/demo security, implement RBAC scaffolding, audit trails, encryption, and compliance artifacts.

P9-01 Mock RBAC for Demo

- Roles: Role 5
- Summary: Implement a minimal RBAC layer with a hardcoded HR Manager user for the demo
- Sub-tasks:
  - P9-01a Define roles/permissions
  - P9-01b Implement a simple auth filter and access checks
- Deliverables: RBAC mock
- Est: 2h
- Dependencies: Phase 1 foundational
- Acceptance criteria:
  - Only HR Manager can access UI-heavy features; masking enforced

P9-02 Audit Logging enhancements

- Roles: Role 5
- Summary: Ensure audit trails cover ingestion, parsing results, AI scoring, overrides, and template changes
- Sub-tasks:
  - P9-02a Extend AuditLog with actions and targetId
  - P9-02b Instrument key flows to emit audit entries
- Deliverables: Audit trails
- Est: 2h
- Dependencies: P2-05, P5-04, P4-03
- Acceptance criteria:
  - Events logged with contextual fields

Phase 10: Testing data and demo prep

Goal: Produce synthetic test data, manage test data lifecycle, and finalize demo assets.

P10-01 Synthetic resumes generator (20+ resumes)

- Roles: Role 4 (Product Lead) + Role 3
- Summary: Generate varied synthetic resumes with edge cases
- Sub-tasks:
  - P10-01a Define schema for synthetic resumes
  - P10-01b Script to generate 20+ resumes
- Deliverables: Test data set
- Est: 3h

- Dependencies: P2-01
- Acceptance criteria:
  - 20+ resumes with diverse profiles present

P10-02 End-to-end test harness

- Roles: Role 5
- Summary: Lightweight test harness for ingestion parsing scoring override
- Sub-tasks:
  - P10-02a Test scenarios
  - P10-02b Basic automation stubs
- Deliverables: Test harness
- Est: 3h
- Dependencies: P2-02, P3-01
- Acceptance criteria:
  - End-to-end path can be exercised in a repeatable manner

P10-03 Demo script, slides, rehearsal

- Roles: Role 4
- Summary: Prepare three demo scenarios; rehearsals
- Sub-tasks:
  - P10-03a Script creation
  - P10-03b Slide deck and recording plan
- Deliverables: Demo materials
- Est: 3h
- Dependencies: P5-02, P7-01
- Acceptance criteria:
  - Demo scenarios are clear, script is rehearsed

Phase 11: Documentation and handoff

P11-01 API specs and developer docs

- Roles: Role 2
- Summary: Document API contracts, data models, prompts, and versioning
- Sub-tasks:
  - P11-01a Generate OpenAPI docs
  - P11-01b Write developer guide
- Deliverables: API docs, README, swagger pages
- Est: 2h
- Dependencies: P1
- Acceptance criteria:
  - Docs accessible and accurate

P11-02 Runbook and rollback plan

- Roles: Role 5
- Summary: Troubleshooting, rollback steps, and post-demo cleanup
- Sub-tasks:
  - P11-02a Write runbook
  - P11-02b Document rollback steps
- Deliverables: Runbook
- Est: 2h
- Dependencies: P9-01, P10-02
- Acceptance criteria:
  - Clear, actionable steps for ops to reproduce and rollback