**WFED 119 Collaborator Job Descriptions and Onboarding v1.1**

*\_Last updated: September 4, 2025\_*

# Changelog v1.1

* Added Section 2: Role Assignments and Workstreams with Lead and Associate for Database and RAG
* Added Section 11: Sprint 0 and Weeks 1-2 Playbook
* Added RACI matrix and monetization alignment checklist

# 1. Purpose and Alignment

This document translates collaborators’ stated preferences and strengths into specific roles, responsibilities, and near term deliverables for the WFED 119 integrated services platform. It aligns work with user facing services and the foundation layer so that student outcomes and course learning objectives are directly supported.

* Platform Services:
* Foundation Layer: Intelligent data management with Database, RAG, and HILT capabilities that supports all services.
* Mission and Vision Architect: Values exploration, mission drafting, and vision board creation, informed by strengths data.
* Wellness Goal Platform: SWOT builder, SMART goal constructor, and progress dashboard with RAG powered insights and HILT goal optimization.
* Action Learning Hub: Team collaboration space with action learning templates, evidence collection, and RAG enhanced problem solving.

Course outcomes linkage: Work streams map to WFED 119 CLOs in self discovery and strengths, mission and agency, and goal setting with implementation.

# 2. Role Assignments and Workstreams

To align responsibilities with overall project flow and monetization, we establish Lead and Associate ownership for two critical workstreams.

## 2.1 Workstream A - Database Management Schema and Migrations

Lead: Jonathan Alavez

Associate: Trivikram Sunil

* Scope:
* Relational schema for users, courses, services, documents, vector metadata, and interaction logs.
* Migrations, seed scripts, rollback plans, backup and restore runbooks.
* Row level security, role based access, audit tables, and data retention policy.
* Data services that expose clean read models to API and RAG layers.
* Phase Deliverables:
* Foundation: ERD v1, schema DDL v1, migration scripts v1, daily snapshot backups, restore test.
* Integration: schema v2 with vector metadata tables, seed loaders for WFED 119, RLS policies, service specific views.
* Launch: capacity plan, index tuning, partitioning plan if needed, production backup policy and recovery time objectives.
* Acceptance Criteria:
* All DDL changes are versioned and reversible.
* Backups validate with a documented recovery test.
* RLS and audit tables cover student and educator scopes.
* API and RAG services can query through read models without ad hoc SQL.
* Risks and Mitigations:
* Scope creep on schema complexity. Mitigate with domain boundaries and documented views.
* Data quality issues. Mitigate with seed validators and CI checks for constraints.

## 2.2 Workstream B - RAG System Ingestion, Embeddings, Retrieval, and Citations

Lead: Trivikram Sunil

Associate: Jonathan Alavez

* Scope:
* Ingestion pipelines for course materials and research artifacts with chunking and metadata policy.
* Embedding registry with model versioning and refresh protocol.
* Retrieval and reranking service, grounding rules, and safe citation format.
* Evaluation harness with relevance, groundedness, and latency metrics.
* Phase Deliverables:
* Foundation: ingestion v1, embeddings v1, search endpoint v1 with top k and reranker, baseline eval set.
* Integration: citation renderer, source attributions, negative sampling, weekly eval dashboard.
* Launch: query taxonomy, guardrails for off topic prompts, cost and latency budgets.
* Acceptance Criteria:
* Ingestion jobs are idempotent and recover from failures.
* Each answer includes machine readable citations that resolve to stored sources.
* Retrieval quality meets thresholds on relevance and groundedness on the evaluation set.
* P50 and P95 latency within agreed budgets under expected load.
* Risks and Mitigations:
* Model drift or embedding mismatch. Mitigate with version pins and periodic reindexing plan.
* Citation brittleness. Mitigate with deterministic renderer and source checksum checks.

## 2.3 Handoffs and Interfaces

* DB read models feed retrieval indexers and evaluation harness.
* RAG service contracts are documented in OpenAPI and validated in CI against DB views.
* Observability spans both workstreams with shared labels for traceability.

## 2.4 RACI Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Workstream | Task | Lead | Associate | Consulted | Informed |
| Database | Schema design and ERD | Jonathan | Trivikram | Hosung | Dr. Yoon |
| Database | Migrations and backups | Jonathan | Trivikram | Hosung | Team |
| Database | RLS and audit policy | Jonathan | Trivikram | Hosung | Team |
| RAG | Ingestion and chunking | Trivikram | Jonathan | Hosung | Team |
| RAG | Embeddings and reindexing | Trivikram | Jonathan | Hosung | Team |
| RAG | Retrieval, rerank, citations | Trivikram | Jonathan | Hosung | Team |
| Shared | Telemetry and evaluation | Trivikram | Jonathan | Hosung | Team |

## 2.5 Monetization Alignment and Document Structure

The project objective is to productize the WFED 119 platform for paid pilots and course licenses. The document is structured so that each workstream produces artifacts required for monetization readiness.

* North Star and KPIs:
* Paid pilot ready by end of Launch phase.
* Improve student task completion rate and educator time savings with measurable deltas.
* Reliability targets: 99.5 percent uptime, P95 response time under 2 seconds for core queries.
* Monetization Readiness Checklist:
* Database: privacy policy mapping, RLS verified, backup and recovery tested, export pipeline for customer data requests.
* RAG: citation compliance, restricted content policy, eval dashboard with exportable reports, cost per query budget.
* Security and compliance: basic access logging, audit trail coverage, secrets rotation policy.
* Packaging: service tier definitions, license terms draft, support and SLA notes.
* Document Map:
* Executive summary and monetization goal.
* Role assignments and workstreams with Leads and Associates.
* Responsibilities by service for Foundation, Mission and Vision, Wellness, and Action Learning.
* Milestones and acceptance criteria mapped to monetization readiness.
* Operating rhythm and review checkpoints.
* Appendices for API contracts, ERD, RAG evaluation rubric, and runbooks.

# 3. Responsibilities by Service

## 3.1 Foundation Layer

* Multi tenant PostgreSQL with service specific extensions and audit trails.
* Vector collections for course content, research, and workflow artifacts.
* Cross service APIs for import, export, and progress sync.
* HILT data structures for interaction logging, triage queues, and consensus.

## 3.2 Mission and Vision Architect

* Import strengths data from LifeCraft Bot.
* Values Exploration Engine conversation flows and card sorting artifacts.
* Mission Statement Builder with alignment validation.
* Vision Board Creator with domain mapping.

## 3.3 Wellness Goal Platform

* Interactive SWOT analysis with AI suggestions.
* SMART goal constructor with validation and milestone generation.
* Progress tracking dashboard with risk assessment and HILT goal optimization.

## 3.4 Action Learning Hub

* Team workspace with action learning forms and evidence collection.
* RAG enhanced problem solving using methodology and case libraries.
* Impact measurement and portfolio generation.

# 4. Milestones and Timeline

* Phase 1 - Foundation Weeks 1 to 4: Shared auth and unified schema, cross service API skeleton, RAG ingestion v0, embeddings v0, search endpoint v0, CI and CD with tests and security checks.
* Phase 2 - Integration Weeks 5 to 8: Data import and export across services, unified dashboard and course alignment testing, beta student workflows for values, goals, and action learning.
* Phase 3 - Production Weeks 9 to 12: Full deployment with performance tuning, educator analytics and outcome reporting, HILT optimization for guidance quality.

# 5. Definition of Done

* Code and tests merged to main with passing CI.
* API contract and usage notes documented.
* Telemetry added for usage, latency, and errors.
* A short demo or screenshot attached in the weekly update.

# 6. Repository and Workflow Conventions

* Monorepo top level apps and packages documented in the project structure.
* Branch naming: feature/<service>-<short-desc> and fix/<scope>-<short-desc>.
* Commit style: Conventional Commits.
* Environments: dev, staging, prod.
* Secrets management documented in the deployment notes.

# 7. Access, Tools, and Standards

* Front end: Next.js, React, TypeScript, Tailwind, shadcn UI, TanStack Query, Zustand.
* Backend: FastAPI or Node with Express or Nest, REST, limited GraphQL.
* Data: PostgreSQL with Prisma or SQLAlchemy, Redis for cache and queues.
* Vector: Qdrant or Pinecone, optional FAISS for offline builds.
* AI: OpenAI compatible APIs or local models, embedding models for indexing.
* Observability: OpenTelemetry with standard labels for service, route, and status.
* CI and CD: GitHub Actions with tests, type checking, and security scans.

# 8. Weekly Operating Rhythm

* Wednesday 10:00 AM ET: Engineering standup with brief demos.
* End of week: written update with status vs plan, risks, and links to artifacts.
* Ad hoc: pairing or code review sessions as needed.

# 9. Open Questions

* Confirm database lead ownership and schedule.
* Confirm priorities for Mission and Vision Architect versus Wellness Goal Platform for the next two sprints.
* Identify initial HILT feedback surfaces for values, goal wizard, and action learning forms.

# 10. Appendix A - Role Reference

* Backend API Engineer: API contracts, auth and RBAC integration, error handling, and pagination.
* Database Lead: Schema, migrations, backups, recovery, and data services for RAG and HILT.
* RAG Engineer: Ingestion, indexing, retrieval, reranking, evaluation, and monitoring.
* HILT Engineer: Feedback capture, annotation, triage and consensus, dashboards, and quality proposals.
* DevOps and Observability: CI and CD, environment configs, telemetry, performance profiling, and cost monitoring.
* Analytics and Research: Outcome measurement frameworks, dashboards, and effectiveness studies.

# 11. Sprint 0 and Weeks 1-2 Playbook

Purpose: Give undergraduate interns clear, small, testable tasks and a streamlined rhythm that produces visible progress toward monetization readiness.

* Cadence:
* Weekly sync: Wednesday 10:00 AM ET. Dr. Yoon may join depending on schedule.
* Pairing: two 30 minute sessions per week. Alternate driver and navigator.
* WIP limit: one ticket per person.
* Git and PR rules:
* Branch names: feature/<workstream>-<short-desc> or fix/<scope>-<short-desc>.
* One ticket per PR. Include a short demo note and one screenshot or a 60 to 90 second clip.
* Lead reviews Associate. Associate reviews Lead.

## Meeting 1 - Agenda and Exit Criteria

* Confirm roles and owners for the two workstreams.
* Set up tools live: clone repo, start containers, run tests.
* Thin vertical slice: one DB migration applied and one document ingested and embedded.
* Assign first tickets with acceptance criteria and due dates.
* Exit criteria:
* Local docker compose up runs Postgres and the minimal API.
* .env.example updated. No secrets in repo.
* One migration applied and rolled back locally.
* One document ingested, chunked, embedded, and written to a local index.
* First two tickets per person are in the tracker and marked In progress.

## Meeting 2 - Agenda and Exit Criteria

* Demos - DB: ERD v0 plus a reversible migration.
* Demos - RAG: ingestion v0 plus search returning top 3 with simple citations.
* Guardrails: backup and restore smoke test scripted, eval harness prints P at 1, P at 3, and MRR for a 20 query set, lint and format checks run in CI.
* Exit criteria:
* Nightly pg\_dump works and restore has been validated.
* Retrieval returns 3 hits and renders citations with source and page.
* Evaluation report checked into repo under /docs/eval.

# 12. Starter Backlog - Intern Scope

This section lists bite sized, testable tasks for the first two weeks. Use the issue template and keep one ticket per PR.

## Database Workstream

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Title | Output | Acceptance |
| DB-01 | Local Postgres in Docker | `docker-compose.yml`, named volume, healthcheck, `init.sql` | Container starts, healthcheck passes, local CLI connects |
| DB-02 | ERD v0 and core tables | ERD image, DDL for users, courses, enrollments, documents, artifacts, interactions | Tables create and sample rows insert |
| DB-03 | Migrations tool | One forward migration and one rollback using Prisma or Alembic | Migrate up and down succeed locally |
| DB-04 | Seed script | Script seeds 10 users, 2 courses, 50 documents, 100 interactions | Row counts match expected numbers |
| DB-05 | Read models for RAG | SQL views that expose clean fields for indexer | RAG job reads from views without ad hoc SQL |
| DB-06 | Backups and restore runbook | Nightly backup script and documented restore steps | Restore validated with a 1 minute smoke test |
| DB-07 | Row level security starter | RLS for student and educator on interactions | Student role cannot read other students’ rows |

## RAG Workstream

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Title | Output | Acceptance |
| RAG-01 | Ingestion v0 | Script reads PDF and MD, normalizes text, chunks around 500 tokens | Serialized chunks with doc id and chunk id exist on disk or DB |
| RAG-02 | Embeddings v0 | Function embeds chunks with version recorded | Embeddings persisted and linked to chunk id |
| RAG-03 | Vector index v0 | Qdrant in Docker or FAISS on disk with build and search helpers | Sample query returns top k with nonzero scores |
| RAG-04 | Retrieval API v0 | /search endpoint returning doc id, chunk id, score | Curl call returns JSON under a small latency budget |
| RAG-05 | Citations and renderer | Citation objects with source, page, snippet and simple renderer | Each result shows a human checkable citation line |
| RAG-06 | Mini evaluation harness | 20 query set with expected sources, compute precision at k and MRR | Markdown report with P at 1, P at 3, MRR |
| RAG-07 | Latency and cost counters | Timing logs and estimated cost per query | Weekly note includes P50, P95, cost per 100 queries |

## Assignment

Database Lead: Jonathan. Associate: Trivikram.

RAG Lead: Trivikram. Associate: Jonathan.

Limit 1 active ticket per person. Pair twice per week for 30 minutes.

## Status labels

* type:setup, type:feature, type:quality
* priority:P1 for Week 1 items, priority:P2 for Week 2 items
* workstream:database, workstream:rag

## Issue Template

Title: <DB-01 Local Postgres in Docker>  
Labels: workstream:database, type:setup, priority:P1, intern  
Owner: @<name>  
Estimate: <hours>  
Definition of Ready:  
- Inputs listed and available  
- Acceptance criteria clear  
Definition of Done:  
- Code merged to main  
- Tests and lint pass in CI  
- Demo note and screenshot linked  
Tasks:  
- [ ] Step 1  
- [ ] Step 2  
Links:  
- PR:  
- Demo:  
- Docs:

## Weekly Status Template

Week of: YYYY-MM-DD  
Planned:  
Done:  
Demos:  
Blockers:  
Next:  
Metrics:  
- Retrieval: P@1, P@3, MRR  
- Latency: P50, P95  
- DB: backup restore test status