**0 Quick-Look Summary (Revised)**

**What you’re building**  
**Developer Guide: AI-Powered Assignment Feedback Tool**

**Project Overview**

Welcome to the team! You are tasked with building a **secure, standalone web application** that will serve as an AI-Powered Assignment Feedback Tool. This tool will integrate with our university's existing Learning Management System (LMS). Its core functions are:

1. Allowing students (redirected from the LMS) to submit specific assignments.
2. Providing students with real-time or near real-time feedback on their submissions, generated by a Generative AI.
3. Tracking submission status, AI feedback, and preliminary assessments in a dedicated database.
4. Enabling instructors to access this application to monitor student progress, review AI-generated feedback, and oversee the assignments processed through this tool.

Essentially, you're creating a specialized "smart drop-box" that enhances our current LMS capabilities by adding an AI feedback layer and a focused progress tracking system for designated assignments.

**Core Development Principles**

* **Security First:** All student data, submissions, and feedback are considered sensitive. Security measures must be robust and implemented from the outset.
* **Privacy by Design:** Proactively embed privacy considerations into the design and functionality, especially concerning AI interactions and data handling.
* **Seamless LMS Integration:** The transition for users (students and instructors) from our main LMS to this tool should be as smooth and secure as possible.
* **Actionable AI Feedback:** The feedback generated by the AI should be clear, constructive, and genuinely helpful for student learning.
* **Iterative Development:** We will build this application in phases, starting with a foundational framework and incrementally adding features based on priority.

**Why this portal (rationale)**

* **Speed to pilot:** A light link-out model avoids LMS provisioning delays yet keeps the door open for future LTI.
* **Data safeguards:** External SSO means no password database to defend; all submissions stored in an object store with server-side encryption.
* **Lowest lift for instructors:** CSV export supports any grade-book while we evaluate direct grade pass-back.

**1 Core Use Cases (must never break) – with “WHY”**

| **#** | **Use case** | **Why it matters** |
| --- | --- | --- |
| 1 | Learner uploads or pastes work before the due date | Core value prop; submission window enforces academic policy |
| 2 | Portal triggers AI feedback within 60 s | Fast turnaround keeps students engaged and guards against duplicate retries |
| 3 | Learner sees feedback history & can resubmit until due | Supports iterative learning; aligns with MIT Open Learning pedagogy |
| 4 | Instructor dashboard lists latest score / version | Rapid triage of who’s stuck; avoids LMS grade-book clutter |
| 5 | Instructor exports CSV of grades + rubrics | Works even if LMS API access is delayed; faculty can reconcile offline |

**2 High-Level Architecture (with rationale)**

Browser ─► Portal (React SPA / Next.js) ─► REST API (FastAPI / Express) ─► DB (PostgreSQL)

│ │

▼ ▼

Object Store (S3-compatible) Worker Queue (Celery / BullMQ) ─► LLM Service

* **React SPA / Next.js:** Mature ecosystem, easy auth SDKs. Next.js enables SSR if SEO or pre-render becomes important but plain CRA/Vite is fine.
* **REST API (FastAPI / Express):** Both have async I/O, good OpenAPI generation, and thriving communities. Pick the one your team supports best.
* **PostgreSQL:** ACID reliability and JSONB for rubric blobs, but any ANSI-SQL DB you already host (e.g., MySQL) is acceptable.
* **S3-compatible store:** Cheap, versioned, and can be swapped (AWS S3, MinIO, Azure Blob).
* **Worker queue:** Decouples AI calls so web tier stays responsive; Celery (Python) or BullMQ (Node).
* **LLM service (pluggable):** Default reference is Azure OpenAI GPT-4o **or an equivalent 32k-context model**(OpenAI direct, Anthropic Claude 3, Mistral-Large, local vLLM). Provide an interface layer so switching providers is a config change, not a refactor.

**3 Tech Stack — Required vs Flexible**

| **Layer** | **Mandatory** | **Flexible options** | **Why** |
| --- | --- | --- | --- |
| Auth | External IdP (OIDC/SAML) & JWT session | Keycloak, Okta, Auth0, your campus IdP | Off-loads MFA & password resets |
| Front-end | React 18+ | Next.js 14, Vite+CRA, Remix | Shared component libraries; React skillsets common on campus |
| API | Any async HTTP framework | FastAPI (Python) **or** Express 5 (Node) | Both generate OpenAPI spec for future integrations |
| DB | Relational SQL | PostgreSQL 15+, MySQL 8+, Supabase | Strong consistency for enrolment checks |
| Object storage | S3 API | AWS S3, Wasabi, MinIO, Azure Blob (s3proxy) | Swappable by URL + credentials |
| LLM | Gemini 2.5 Flash Preview 04-17 |  |  |
| Queue | Any durable task queue | Celery 5, BullMQ, Rabbit with workers | Isolates long-running AI calls |

**Guiding principle:** Default picks are “known-good.” Replace any **flexible** element if you (a) already operate it in production or (b) need on-prem hosting for policy reasons. Update the .env.example and README when you diverge.

**4 Phase Plan (iterate by importance)**

| **Phase** | **Deliverable** | **Why first?** | **Est. Order** |
| --- | --- | --- | --- |
| 0 | Repo skeleton, CI, IaC, health check route | Sets guard-rails; dev starts in a clean main branch | 0–1 |
| 1 | Secure launch link → upload → DB write | Proves end-to-end flow without AI; de-risks SSO | 1–2 |
| 2 | Worker queue + LLM call → feedback stored | Unlocks learning value; isolates model latency | 2–4 |
| 3 | Learner dashboard & resubmission | User-facing polish; drives demo credibility | 4–5 |
| 4 | Instructor CSV export & basic analytics | Minimum reporting to sunset Google Sheets | 5–6 |

**5 Key Implementation Hints (selected)**

1. **Signed link util** – keep the secret in AWS SSM or Vault. Rotate every term.
2. **AI feedback schema** – store both *raw LLM JSON* and a *validated projection* so future rubric mapping is painless.
3. **Explainability** – include usage.tokens and model\_name in ai\_conf for cost audits.
4. **Swappable LLM adapter** – define an abstract ai\_provider.generate(prompt, settings); each provider lives in adapters/.
5. **Why adapter?** Prevents vendor lock-in and lets you A/B models for quality vs price.

**End of document**