**SRS**

**سیستم مدیریت و تحلیل داده‌های دانشگاهی  
 (Academic Data Platform)**

**نسخه** 1.0 :   
**تاریخ**  2025-09-15 :   
**نویسنده :** غریب آشنا  
**مرجع‌ها :**  RFP کامل پروژه، فایل‌های UML در Canvas، مستندات پیشنهادی معماری

**مقدمه**

**هدف سند**

این سند نیازمندی‌های نرم‌افزاری سیستم مدیریت و تحلیل داده‌های دانشگاهی را به‌صورت رسمی و قابل‌استناد تعریف می‌کند. هدف فراهم‌کردن مبنای مشترک بین ذی‌نفعان برای طراحی، پیاده‌سازی، تست و ارزیابی سیستم است.

**دامنه ی پروژه**

پلتفرم شامل ماژول‌های مدیریت کاربران/دانشجویان/اساتید، پردازش و ذخیره‌سازی داده‌ها، ماژول زمان‌بندی (scheduling)، تحلیل‌های یادگیری ماشین (پیش‌بینی دانشجوی در ریسک)، سیستم توصیه‌گر، داشبورد مدیریتی و APIهای یکپارچه‌سازی با سیستم‌های دانشگاهی است.

**مخاطبان سند**

* مدیر پروژه PO
* تیم توسعه (Backend, Frontend, ML, DevOps)
* QAتسترها
* نمایندگان دپارتمان اساتید
* حامیان زیرساختی IT دانشگاه

**تعریف‌ها و اختصارات (خلاصه)**

* SRS: Software Requirements Specification
* API: Application Programming Interface
* LMS: Learning Management System   
  مثلاً Moodle/Canvas
* SIS: Student Information System
* ML: Machine Learning
* KPI: Key Performance Indicator

**معرفی کلی سیستم**

**چشم‌انداز محصول**

یک پلتفرم سازمانی-آکادمیک است که داده‌های آموزشی و پژوهشی را جمع‌آوری، پردازش، تحلیل و نتیجه را عبر داشبورد و API ارائه می‌دهد. اهداف شامل بهبود تصمیم‌گیری آموزشی، پیش‌بینی ریسک افت تحصیلی و تسهیل پژوهش دانشگاهی است.

**ویژگی‌های کلیدی (خلاصه)**

* مدیریت کاربران و نقش‌ها (دانشجو، استاد، مدیر)
* ثبت‌نام/نمره‌دهی/ترنسکریپت
* زمان‌بندی کلاس‌ها با بهینه‌سازی (ILP/GA)
* ML Pipelines برای پیش‌بینی ریسک و Recommendation
* داشبورد مدیریتی و گزارش‌سازی
* APIهای امن برای یکپارچه‌سازی با LMS/SIS
* قابلیت مقیاس‌پذیری و استقرار در Kubernetes

**مدل‌های کاربری (Actors)**

* Student دانشجو
* Faculty استاد
* ) Administrator ادمین دانشگاه(IT ,
* Department Officer معاون یا مسئول گروه
* Analytics Engineer / ML Engineer
* External Systems (LMS, SIS, Library)

**نیازمندی‌های کاربردی (Functional Requirements)**

هر نیاز با یک شناسه (FR-xxx) مشخص شده است.

جدول خلاصه: برای خوانایی ابتدا فهرست کلی ماژول‌ها، سپس شرح جزئی هر نیاز.

**ماژول Authentication & User Management**

**FR-AUTH-001**   
 ثبت‌نام و ورود کاربران

* **شرح:** سیستم باید امکان ثبت‌نام کاربران (دانشجو/استاد/ادمین) و ورود امن با JWT/OAuth2 را داشته باشد.
* **ورودی** username/email, password, role**:**
* **خروجی**  access\_token, refresh\_token **:**
* **پیش‌شرط :** هیچ
* **پس‌شرط:** کاربر ایجاد شده در جدول users ثبت می‌شود.
* **اولویت :** حیاتی
* **معیار پذیرش:** کاربر بتواند با username/pass وارد شود و دریافت توکن معتبر.
* **API نمونه**  POST /auth/register, POST /auth/login **:**

**FR-AUTH-002**

مدیریت نقش‌ها و سطوح دسترسی

* اجازهی تعریف نقش سفارشی، assign کردن نقش به کاربر، کنترل دسترسی براساس role/permissions.
* **Acceptance:** RBAC enforce شده و endpointهای حساس فقط با نقش مجاز اجرا شوند.

**FR-AUTH-003** — Single Sign-On ( اختیاری/اولویت متوسط )

* ادغام با Keycloak/LDAP/SSO دانشگاه.

**ماژول Student / Faculty Data**

**FR-STU-001** — CRUD پروفایل دانشجو

* نمایش ترنسکریپت، سوابق ثبت‌نام، وضعیت فعلی.
* **API:** GET /students/{id}, PUT /students/{id}
* **Acceptance :**  داده‌ها مطابق DDL ذخیره و سازگار نمایش داده شود.

**FR-FAC-001** — CRUD پروفایل استاد

* مدیریت اطلاعات پژوهشی، انتشارات و حوزهی تخصصی.

**Courses, Sections & Enrollment**

**FR-COURSE-001** مدیریت Course/Section

* ایجاد، ویرایش، حذف درس و سکشن برای ترم‌ها.
* **API:** POST /courses, POST /sections

**FR-ENROLL-001** ثبت‌نام دانشجو در Section

* بررسی پیش‌نیاز (prerequisites) و ظرفیت، ثبت‌نام و انتشار رویداد EnrollmentCreated در کافکا.
* **Acceptance:** در صورت پر بودن capacity، ثبت‌نام رد شود و پیام خطای معنی‌دار بازگردد.

**FR-GRADE-001** ثبت نمره

* اساتید بتوانند نمره اختصاص دهند و GradeUpdated منتشر شود.
* **API:** POST /grades

**Scheduling زمان‌بندی کلاس‌ها**

**FR-SCH-001** تولید برنامهی زمان‌بندی

* **شرح:** سرویس scheduling باید با دریافت constraints (rooms, instructors availability, capacity, course conflicts) یک schedule تولید کند.
* **ورودی** term, year, list of course sections, constraints JSON**:**
* **خروجی** schedule\_id **:**  و ساختار زمان‌بندی قابل نمایش
* **معیار پذیرش:** خروجی قابل نمایش در UI و بدون) conflict تا حدِ امکان(

**FR-SCH-002** ویرایش دستی زمان‌بندی و override rules

* مدیر بتواند تغییرات دستی اعمال کند و سیستم consistency check انجام دهد.

**Analytics — At-Risk Prediction**

**FR-AN-001** اجرای job پیش‌بینی ریسک

* **شرح** AnalyticsService **:**  باید بتواند یک job آموزش/پیش‌بینی اجرا کند (batch یا incremental) و risk\_scores را در جدول risk\_scores ذخیره کند.
* **API:** POST /analytics/at\_risk/run
* **معیار پذیرش:** تولید جدول risk\_scores برای داَتاست نمونه، متریک AUC گزارش شود.

**FR-AN-002** نمایش گزارش در داشبورد

* گزارش با امکان فیلتر بر حسب program, year, course.

**Recommendation Engine**

**FR-REC-001** توصیه ی دروس به دانشجو

* براساس تاریخچهی نمرات، علاقه‌مندی‌ها و هم‌فرض‌های collaborative filtering توصیه کند.
* **API:** GET /recommendations/{student\_id}

**Research Management**

**FR-RES-001**  مدیریت انتشارات و شبکهی همکاری

* وارد کردن ResearchPaper و نگاشت به اساتید، ساخت گراف همکاری (edge weights).
* **Feature:** Export collaboration graph for network analysis.

**Reporting & Export**

**FR-REP-001** تولید گزارش PDF/Excel

* Scheduled reports: monthly/termly for department heads.

**Integration APIs**

**FR-INT-001** — Connector به LMS (Moodle/Canvas)

* استخراج دادهی فعالیت دانشجو، حضور، assignment grades.
* **FR-INT-002** — Connector به SIS: دانشجویان و نمرات پایه را سینک کند.

**نیازمندی‌های غیرکاربردی (Non-Functional Requirements)**

هر NFR مختص با شناسه.

**عملکرد و مقیاس‌پذیری**

* **NFR-PERF-001:** پاسخ API خواندن پروفایل (p50) ≤ 150ms در حالت cache cold ، (p95) ≤ 500ms برای بار تا 1000 concurrent users.
* **NFR-PERF-002:** سیستم باید قابل مقیاس تا 10k active users/day با افزایش افقی در Kubernetes باشد.
* **NFR-DATA-001:** قابلیت دسته‌بندی و پردازش batch داده تا 1M records در روز برای ETL.

**در دسترس‌پذیری و پایداری**

* **NFR-AVAIL-001:** SLA 99.5% برای APIهای اصلی در production.
* **NFR-BDR-001:** RTO ≤ 1 hour، RPO ≤ 1 hour برای داده‌های سیستمی (نمرات و ثبت‌نام). (برای داده‌های تاریخی RPO ≤ 24h)

**امنیت**

* **NFR-SEC-001:** TLS 1.2+ برای کلیه ارتباطات.
* **NFR-SEC-002:** احراز هویت مبتنی بر OAuth2/JWT، RBAC، و رول‌های دسترسی.
* **NFR-SEC-003:** PII در DB باید رمزنگاری at-rest و at-transit.
* **NFR-SEC-004:** Audit logging برای عملیات حساس (grade changes, enrollments changes).

**حریم خصوصی و تبعیت قانونی**

* **NFR-PRIV-001:** پشتیبانی از anonymization و export data requests مطابق GDPR-like policies
* **NFR-PRIV-002:** نگهداری داده‌های حساس (مثل شماره ملی) به‌صورت ایمن و با دسترسی محدود.

**قابلیت نگهداری و توسعه‌پذیری**

* **NFR-MNT-001:** ماژول‌ها باید با contracts مشخص (OpenAPI) توسعه‌پذیر باشند.
* **NFR-MNT-002:** اتصال CI/CD با تست‌های unit/integration و lint.

**قابل مشاهده‌بودن (Observability)**

* **NFR-OBS-001:** هر سرویس باید metric (Prometheus) و لوگ (ELK) صادر کند.
* **NFR-OBS-002:** داشبورد وضعیت سیستم و alerting برای خطاهای بحرانی.

**UX / Accessibility**

* **NFR-UX-001:** UI باید ریسپانسیو و قابل‌دسترسی (WCAG AA) باشد.

**سناریوهای کاری (Use Cases) — به‌صورت دقیق**

برای هر سناریو: Actor، هدف، جریان اصلی، جریان استثنا، پیش‌شرط، پس‌شرط، APIهای مرتبط.

**UC-01: Login & Session**

* **Actor:** Any user
* **هدف:** احراز هویت و ایجاد session معتبر
* **جریان اصلی:** کاربر credentials می‌دهد → API /auth/login اعتبارسنجی می‌کند → issue access\_token + refresh\_token → client ذخیره و از آن استفاده می‌کند.
* **استثنا:** credentials نادرست → کد 401 و پیام خطا.
* **پیش‌شرط:** کاربر ثبت‌نام شده باشد.
* **پس‌شرط:** token با expiry صادر شود.

**UC-02: Student Enrollment**

* **Actor:** Student (via UI)
* **هدف:** ثبت‌نام در یک Section
* **جریان اصلی:** Student درخواست ثبت‌نام → Enrollment Service بررسی capacity و prereq → در صورت قبول ثبت‌نام و پیام EnrollmentCreated در Kafka منتشر می‌شود → DB update → notify student.
* **استثنا:** capacity full یا prereq missing → نمایش خطای مناسب.
* **API:** POST /enrollments
* **Acceptance:** enrollment در DB ذخیره و event منتشر شده باشد.

**UC-03: Instructor grade submission**

* **Actor:** Faculty
* **هدف:** ثبت نمره برای یک دانشجو
* **جریان:** Instructor باز می‌شود → انتخاب course section → وارد کردن grade → grade saved & GradeUpdated event published → triggers analytics recompute (اختیاری).
* **API:** POST /grades

**UC-04: Run At-Risk Prediction (Batch)**

* **Actor:** Admin/Analytics Engineer (or scheduled cron)
* **هدف:** اجرای job برای محاسبه risk\_scores برای مجموعه‌ای از دانشجویان
* **جریان:** request to /analytics/at\_risk/run → AnalyticsService queries feature store → submit training/inference job to ML Orchestrator → on completion store risk\_scores → publish report → notify admin.
* **Postcondition:** risk\_scores table populated, dashboard updated.

**UC-05: Generate Schedule**

* **Actor:** Department Officer (or automated)
* **هدف:** تولید یک schedule conflict-free تا حد ممکن
* **جریان:** send constraints → SchedulingService uses optimizer (ILP/GA) → returns schedule\_id + details → persisted → UI displays Gantt-like schedule.
* **Acceptance:** conflict metrics reported (zero conflicts or minimal).

**User Stories (Agile-ready)**

هر استوری همراه با معیار پذیرش (AC).

**US-001 (Priority: High)**

**As a** student, **I want** to view my transcript and enrolled courses, **so that** I can track progress.  
**AC:** GET /students/{id} returns courses, grades, GPA; UI renders table and export to PDF.

**US-002 (High)**

**As an** instructor, **I want** to submit grades for my section, **so that** students' records update immediately.  
**AC:** POST /grades accepts array of {enrollment\_id, grade}; returns success and triggers GradeUpdated events.

**US-003 (High)**

**As an** admin, **I want** to run at-risk prediction for a program, **so that** we can proactively support students.  
**AC:** POST /analytics/at\_risk/run with program\_id returns job\_id; when job finishes risk\_scores saved; admin dashboard shows list.

(و ادامه: US-004 scheduling, US-005 recommendations، US-006 admin reports، US-007 SSO integration، US-008 export data request)

**7. Data Requirements & Data Dictionary (نمونه)**

**7.1 جدول کاربران (users)**

* **user\_id (UUID)** — PK
* **username (TEXT)** — unique
* **email (TEXT)** — unique, indexed
* **role (TEXT)** — enum [student, faculty, admin, staff]
* **password\_hash (TEXT)** — bcrypt

**7.2 جدول students**

* **student\_id (UUID)** — PK
* **user\_id (UUID)** — FK→users.user\_id
* **admission\_year (INT)**
* **program\_id (UUID)** — FK→programs.program\_id
* **status (TEXT)** — active/suspended/graduated

**7.3 risk\_scores**

* **risk\_id, student\_id, model\_id, risk\_score, computed\_at**

برای هر جدول اصلی، نوع، محدوده، و نمونه داده لازم است در فایل DDL قرار گیرد (نسخه کامل قبلاً ساخته شده؛ نسخهٔ نهایی را در فایل schema.sql قرار می‌دهم).

**7.4 حجم و رشد داده (پیش‌بینی)**

* کاربران: 50k (تا 5 سال)
* رویدادهای ثبت‌نام: 1M/year
* لاگ‌ها: 10M/year
* feature store: ~50 GB اولیه، رشد ماهانه 5–10 GB

**8. External Interfaces & Integration**

* **LMS (Moodle/Canvas):** API polling or webhook ingestion for assignment grades and activity logs.
* **SIS:** CSV/REST sync for student master data.
* **LDAP/SSO:** برای احراز هویت و Single Sign-On.
* **Email/SMS Gateway:** برای notifications (SMTP, Twilio).
* **Object Storage (S3):** ذخیره Parquet/feature files.
* **Message Broker (Kafka/RabbitMQ):** event-driven processing.
* **GPU Cluster / Kubernetes:** اجرای training jobs.

**API Contracts:** تمامی سرویس‌ها با OpenAPI spec تعریف می‌شوند؛ نسخه‌بندی (v1, v2) رعایت شود.

**9. Security, Privacy & Compliance (جزئیات)**

**9.1 تهدیدات اصلی (Threat Model)**

* سرقت توکن‌های JWT → mitigation: short-lived access tokens + refresh rotation + token revocation list
* SQL Injection → mitigation: استفاده از ORM/parametrized queries
* Data exfiltration → mitigation: DLP, encryption at rest, IAM policies

**9.2 سیاست‌های دسترسی**

* Role-based access (student only own data; instructor data of courses they teach; admin full access)
* Admin actions (grade override) require MFA & audit logging

**9.3 نگهداری داده شخصی**

* PII encrypted at rest
* Data access logs exported to ELK
* Data retention policy: student records permanent; logs rotated after 2 years (configurable)

**10. Test Plan (خلاصه)**

**10.1 نوع‌های تست**

* Unit tests for each service/component
* Integration tests for API contracts (using test DB)
* End-to-end tests for critical flows (enrollment, grade submission, at-risk run)
* Load/performance tests (JMeter / k6) to validate NFR-PERF
* Security tests: static code analysis + pentest (مرحله نهایی)

**10.2 معیارهای پذیرش کلی (Acceptance Criteria)**

* All critical APIs pass functional and integration tests.
* At-risk model AUC ≥ 0.75 on validation data (initial target).
* API latency and availability conform to NFR thresholds.
* CI pipeline green & helm charts deploy cleanly to staging
* .

**11. Traceability Matrix (نمونه خلاصه)**

نشان می‌دهد هر FR به کدام ماژول، تست، و user story مرتبط است.

|  |  |  |  |
| --- | --- | --- | --- |
| Req ID | شرح خلاصه | ماژول | Test Case ID |
| FR-AUTH-001 | Login/Register | Auth Service | TC-AUTH-01 |
| FR-ENROLL-001 | Enrollment | Enrollment Service | TC-ENR-01 |
| FR-AN-001 | At-Risk Run | Analytics Service | TC-AN-01 |

**. Deployment & DevOps Requirements**

* GitHub repo with CI (lint, unit tests)
* Multi-stage Docker images + image scanning
* Kubernetes cluster with namespaces (dev/staging/prod)
* Helm charts for each microservice
* Backups: automated Postgres backups to S3 daily

**13. Constraints, Assumptions & Risks**

**13.1 محدودیت‌ها**

* دسترسی به دادهٔ واقعی دانشگاه ممکن است محدود یا مشروط به قوانین حریم خصوصی باشد.
* منابع GPU محدود ممکن است سرعت تحقیق و آموزش را کم کند.

**13.2 فرض‌ها**

* دانشگاه موافقت به ارائه نمونه داده (anonymized) برای مرحلهٔ توسعه/آزمون خواهد داد.
* تیم حداقلی شامل یک ML engineer خواهد بود.

**13.3 ریسک‌ها و برنامهٔ کاهش**

* ریسک کمبود داده برای ML → mitigation: use public datasets + synthetic data + transfer learning.
* ریسک امنیتی → mitigation: early security review & automated scanning.

**14. معیارهای موفقیت پروژه (KPIs)**

* مدل پیش‌بینی: AUC-ROC ≥ 0.75
* API latencies: p95 ≤ 500ms
* Uptime ≥ 99.5%
* Time-to-deploy (CI->prod) ≤ 1 day
* تعداد کاربران فعال در ماه (MAU) ≥ 1000 در 6 ماه اول (هدف پذیرشی)

**15. خروجی‌های سند و پیوست‌ها**

* فایل schema.sql (نسخهٔ کامل DDL) — موجود
* OpenAPI specs (برای همه سرویس‌ها) — جهت تولید بعدی
* Use-case diagrams, sequence diagrams — موجود در Canvas (PlantUML)
* Test Plan detailed — تولید در sprint planning

**16. گام‌های بعدی پیشنهادی (Immediate Action Items)**

1. تایید SRS توسط ذی‌نفعان دانشگاه (PO + IT + 1 استاد) — **Sign-off**.
2. تهیه dataset نمونه و قرار دادن در S3 (anonymized).
3. ایجاد repo و CI skeleton + قرار دادن schema.sql و seed data.
4. اجرای Sprint 1 (Infra + Auth + DB migrations). (جزئیات در بک‌لاگ موجود است.)

**17. پیوست — مثال API schema برای یکی از endpointها (OpenAPI-like)**

POST /analytics/at\_risk/run

* Request body:

**خلاصهٔ عمل‌گرایانه و برنامهی اجرایی (فقط چند خط)**

* من همین الآن SRS کامل نسخهٔ پایه را تهیه کردم. این سند پایهی رسمی برای شروع توسعه است.
* پیشنهاد می‌کنم **الان** — بدون فوت وقت — SRS را به اساتید نشان دهی برای تایید رسمی (Sign-off). پس از تایید، من **HLA (High-Level Architecture)** و **OpenAPI specs** و سپس **backlog تفصیلی Sprint 1** را تولید می‌کنم و فایل‌های قابل‌اجرا (schema.sql کامل، openapi.yaml) را آماده می‌کنم.