Software Requirements Specification (SRS)

Project: A to Z Academy — 3-Core Platform (Educational · System/Operations · Marketing)

Version: 0.1 (Draft)

Author: Akram (drafted with ChatGPT)

Date: 26 August 2025

---

1. Executive Summary

A to Z Academy needs an integrated platform built around **three cores**:

1. **Educational Core** — course catalog, lessons, assignments, quizzes, progress tracking, grades.
2. **System / Operations Core** — users & roles, authentication, permissions, online homework submission, admin dashboards, notifications, reporting.
3. **Marketing Core** — public-facing site with course promotion, landing pages, testimonials, enrollment funnels, analytics.

The goal is a single system where marketing attracts learners, operations manage users and content, and the educational experience delivers learning and assessment.

---

2. Stakeholders

- **Primary:** Students, Teachers/Instructors, Admins
- **Secondary:** Marketing team, Parents/Guardians, Academy Owner(s)
- **System:** Backend developers, Frontend developers, DevOps, QA

---

3. Scope

In scope

- User registration & login (email/password, optional social auth)
- Role-based access (Student, Teacher, Admin, Marketing)
- Course creation & management (modules, lessons, resources)
- Assignment creation, submission (file/audio), grading and feedback
- Quizzes (auto-graded MCQ + manually graded questions)
- Student progress dashboard and gradebook
- Public marketing site: course pages, hero banners, CTA, contact form
- Enrollment flow and checkout (basic free/paid flag; payment processor placeholder)

- Basic analytics & reports (enrollments, completion rates, active users)
- Notifications (email + in-app)

Out of scope (v0.1)

- Full ecommerce (subscriptions, coupons) — placeholder
- Live video streaming / WebRTC classroom — placeholder
- Advanced proctoring or plagiarism detection — future enhancement

---

4. Definitions and Abbreviations

- **SRS** — Software Requirements Specification
- **Admin** — platform administrator
- **Instructor** — course creator/teacher
- **Student** — enrolled learner

---

5. High-Level Architecture

Three logical layers:

- **Presentation Layer** — public marketing site + logged-in SPA (teacher/student/admin) using templates or modern frontend framework.
- **Application Layer (Backend API)** — REST/GraphQL endpoints, business logic, auth, role checks.
- **Data Layer** — relational database (SQLite for dev / PostgreSQL for prod). Optional JSON fields for flexible content.

Services: notifications worker, scheduled tasks (cron) for reports and reminders, background processing for large uploads.

---

6. Functional Requirements (by Core)

6.1 Educational Core

- FR-E1: Admin/Instructor can create a course with: title, category, short description, full description, image, price (optional), syllabus (modules → lessons).
- FR-E2: Instructor can create lessons containing text, video link, attachments (PDF, audio), and reference files.
- FR-E3: Instructor can create assignments with due date, instructions, allowed file types (audio for recitations), and auto/manual grading options.
- FR-E4: Students can view enrolled courses, access lessons, submit assignments (file upload or audio recording), and view feedback/grade.
- FR-E5: System stores submission history and timestamp; supports resubmission policy (configurable).

- FR-E6: Quiz builder supports multiple choice (single/multiple), short answer (manual grade), and scoring rules.
- FR-E7: Progress tracking: mark lesson complete, percentage/completion badge, course certificates (PDF stub).

6.2 System / Operations Core

- FR-S1: Role-based authentication and authorization (roles: Student, Instructor, Admin, Marketing).
- FR-S2: Admin dashboard: user management (create/disable), course moderation, view reports, system settings.
- FR-S3: Student profile stores name, email, phone, guardian (optional), progress, certificates.
- FR-S4: Assignment queue: teachers see pending submissions, grade/return feedback, export grades as CSV.
- FR-S5: Notifications: system can send emails for assignment due reminders, enrollment confirmations, and announcements.
- FR-S6: Audit logs for critical actions (user creation, role changes, course publish/unpublish).

6.3 Marketing Core

- FR-M1: Public homepage with hero, course highlights, testimonials, and contact form.
- FR-M2: Course landing pages with CTA (Enroll), syllabus preview, instructor bio, and FAQ.
- FR-M3: Simple enrollment funnel: click enroll → register/login → confirm enrollment (free or placeholder paid flow).
- FR-M4: Marketing content manager (role) can create/edit static pages and banners.
- FR-M5: Basic analytics dashboard: pageviews (approx), enrollments per course, source channel (manual tagging).

---

7. Nonfunctional Requirements

- NFR-1: Security: passwords hashed (bcrypt/argon2), HTTPS enforced in production, input validation.
- NFR-2: Performance: average page load < 2s under typical load (baseline for deployment), API responses < 500ms for simple queries.
- NFR-3: Availability: 99% SLA target for core services (post-MVP).
- NFR-4: Scalability: design should allow horizontal scaling (stateless backend + shared DB and object storage for media).
- NFR-5: Maintainability: follow modular code structure, use migrations for DB changes, unit tests for main flows.
- NFR-6: Localization: Arabic & English support for UI strings; default language can be chosen per user.

---

8. Data Model (Tables overview)

Suggested core tables:

- users (id, name, email, password_hash, phone, role, is_active, created_at)
- profiles (user_id, bio, avatar_url, location, extra_json)

- courses (id, title, slug, short_desc, full_desc, image_url, category_id, instructor_id, price, status, created_at)
- modules (id, course_id, title, position)
- lessons (id, module_id, title, content, video_url, attachments_json, position)
- assignments (id, course_id, lesson_id, title, description, due_date, max_score)
- submissions (id, assignment_id, student_id, file_url, audio_url, text, score, feedback, submitted_at)
- quizzes (id, course_id, title, settings_json)
- questions (id, quiz_id, type, text, choices_json, correct_json)
- enrollments (id, course_id, student_id, status, enrolled_at)
- notifications (id, user_id, type, payload_json, read_flag, created_at)
- audit_logs (id, user_id, action, details_json, created_at)

Produce an ERD later (visual) based on the above tables.

---

9. API (Example endpoints)

- `POST /api/auth/register` — register user
- `POST /api/auth/login` — login
- `GET /api/courses/` — list courses (public)
- `GET /api/courses/{slug}` — course details
- `POST /api/courses/` — create course (instructor)
- `POST /api/courses/{id}/enroll` — enroll
- `GET /api/my-courses/` — student enrolled courses
- `POST /api/assignments/{id}/submit` — submit assignment (multipart)
- `GET /api/submissions/pending` — teacher pending submissions
- `POST /api/submissions/{id}/grade` — grade submission

---

10. UI / Pages (minimum)

Public (Marketing)

- Homepage
- Course listing
- Course detail / landing page
- About, Contact, Terms

Authenticated

- Student dashboard (courses, recent activity)
- Course player (lessons + resources)
- Assignment submission page
- Quiz interface
- Instructor dashboard (create course, student list, gradebook)
- Admin dashboard (site settings, user management)

11. Use Cases & User Stories (priority)

- **US-1 (High):** As a Student, I can register and enroll in a course so I can access lessons and submit assignments.
- **US-2 (High):** As an Instructor, I can create a course and publish it so students can enroll.
- **US-3 (High):** As a Teacher, I can view pending assignment submissions, grade them, and provide feedback.
- **US-4 (Medium):** As a Marketing Manager, I can edit the homepage banners and create a course landing page.
- **US-5 (Medium):** As an Admin, I can generate CSV reports of enrollments and export gradebooks.

---

12. Acceptance Criteria (examples)

- Enrollment flow completes and a student appears in the course enrollments table.
- Student can upload an audio file (mp3/wav) $\leq$ 10 MB and it appears in submission preview.
- Instructor can grade a submission and the grade reflects in the student gradebook.

---

13. Security & Compliance

- Enforce HTTPS in production.
- Store passwords with strong one-way hashing.
- Role checks on every restricted API.
- Rate-limit public endpoints (login, register) to prevent abuse.

---

14. Deployment & DevOps

- Development: use SQLite and local filesystem for media.
- Staging & Production: PostgreSQL + S3-compatible storage for media, external SMTP for emails.
- CI: automated tests, migrations run on deploy, static file build.
- Containerization: Dockerfile + compose for local development; Kubernetes or managed service in production.

---

15. Roadmap & MVP (suggested)

Phase 1 (MVP) — Core education features + basic marketing site + role-based auth. **Phase 2** — Payments, certificates, richer analytics, Arabic localization. **Phase 3** — Live classes, advanced reporting, mobile apps.

---

16. Non-technical deliverables

- SRS (this document) in English + Arabic version (upon request)
- ERD diagram (visual)
- Wireframes for primary pages
- Test cases and acceptance tests for each high-priority user story

---

17. Next actions (practical)

1. Confirm core user roles and policies (resubmission, late submission penalty, instructor approval policy).
2. Finalize data retention and privacy policy.
3. Create initial ERD diagram and wireframes for Student Dashboard and Course Landing Page.
4. Prepare backlog (epics → stories → tasks) and an initial sprint plan (2-week sprint recommended).

---

18. Appendices

- Example data size estimates, media storage planning, sample CSV export format, and checklist for release.

End of Draft SRS v0.1