ARCHIFY – أرشيفي

Cahier des charges fonctionnel et technique (FR)

Version 1.0 — 30/09/2025

Auteurs: Proger Tech — Équipe produit et technique

Contact: contact@progertech.com

1) Vision et objectifs

- Vision: Archify est une plateforme universitaire centralisée et moderne pour consulter des cours, vidéos explicatives, fiches PDF et archives d’examens, pensée pour les étudiants francophones (arabe à venir).

- Objectifs clés:

• Améliorer l’accès aux contenus pédagogiques (cours, notes, examens corrigés)

• Accroître l’engagement et la réussite (recommandations, quiz optionnels)

• Offrir un modèle durable via abonnements mensuels/annuels

• Fournir un back-office simple pour l’upload et la gestion de contenus

- Différenciation:

• Expérience mobile-first, simple et élégante

• Organisation claire par semestre, département, matière

• Accès premium sécurisé aux vidéos et PDF

2) Public cible et personas

- Cible: Étudiants universitaires (départements initiaux: notre promotion), francophones. Version arabe statique envisagée.

- Personas principaux:

• Étudiante “Ahmed” (L2 Informatique du Gestion): consulte vidéos, télécharge PDF, recherche les sujets d’examens

• Étudiant “Minetou” (L3 Informatique du Gestion): prépare ses partiels via les archives, tri par professeur

• Admin “Prof/Chargé de TD”: met en ligne des cours, gère corrections, modère

3) Portée (MVP → V1.0 → V2.0)

- MVP (prioritaire):

• Authentification (inscription/connexion, mot de passe oublié)

• Tableau de bord étudiant (cours suivis, progression simple, recommandations basiques)

• Catalogue de cours/leçons (vidéos, PDF, archives d’examens)

• Recherche + filtres (semestre, département, matière, professeur, tri: populaires/récents/examens)

• Abonnements payants (mensuel/annuel) via Bankily/Masrivi/Sedad (gateway local en option)

• Gating premium (accès restreint aux contenus)

• Commentaires (modération basique)

• Admin: upload/gestion contenus, gestion étudiants et abonnements, analytics basiques

- V1.0 (post-MVP proche):

• Quiz optionnels par leçon (QCM)

• Recommandations améliorées (popularité, tags)

• Pages publiques SEO (Angular SSR) pour découverte des cours

• Coupons promo et périodes d’essai

- V2.0 (évolution):

• Recommandations IA (personnalisées)

• Intégration passerelle locale de paiement (si marché)

• App mobile dédiée (si besoin)

4) Exigences fonctionnelles — Détails

4.1 Comptes et rôles

- Utilisateurs: Étudiants (accès standard/premium), Admin, SuperAdmin

- Inscription: email + mot de passe (ou SSO futur). Vérification email (post-MVP si nécessaire)

- Connexion: email + mot de passe, tokens JWT (cookies HttpOnly)

- Mot de passe oublié / réinitialisation par email sécurisé

- Profil: nom, département, semestre, photo (optionnel)

- Rôles et autorisations:

• Étudiant: lecture contenus, commentaires, abonnement

• Admin: CRUD cours/leçons/assets, modération

• SuperAdmin: gestion admins, paramètres globaux

4.2 Cours & leçons

- Organisation: semestre → département → matière → cours → leçons

- Contenu leçon:

• Vidéo: Vimeo (privé; domaine autorisé, player SDK) ou YouTube (non répertoriée)

• PDF: notes de cours, fiches, sujets d’examens (S3/Cloudinary)

• Métadonnées: professeur, durée, tags, niveau, statut (gratuit/premium)

• Statistiques: vues, temps de visionnage (agrégé)

- Fonctionnalités:

• Lecture vidéo intégrée (player moderne, vitesse, qualité)

• Visionneuse PDF (pdf.js/ngx-extended-pdf-viewer)

• Progression simple par leçon (marquer comme “vu”)

• Contenus premium accessibles uniquement aux abonnés actifs

• Protection du contenu: interdiction stricte de capture d’écran et d’enregistrement vidéo; filigrane dynamique (ID utilisateur, email haché, horodatage) superposé; désactivation du clic droit et du téléchargement; désactivation Picture-in-Picture/AirPlay; lecture via URLs signées et sources obfusquées.

4.3 Recherche & filtres

- Recherche mot-clé (titre, description, professeur, tags)

- Filtres: semestre, département, matière, type (vidéo/PDF/examens), statut (gratuit/premium)

- Tri: plus vus, plus récents, orientés examens

- Suggestions: cours populaires, récemment ajoutés

4.4 Abonnements et paiements

- Plans: Mensuel, Annuel (prix définis en Admin)

- Paiements: Bankily/Masrivi/Sedad (intégration REST, webhooks). Passerelle locale (V2.0)

- Accès premium: à l’activation du paiement via webhook vérifié

- Renouvellement, expiration: mise à jour automatique de l’état d’abonnement

- Annulation: fin d’accès à l’expiration de la période payée

- Coupons (V1.0), essai (optionnel)

4.5 Interactions

- Commentaires par leçon: rédaction, suppression (par auteur ou admin), signalement, modération

- Quiz (V1.0): QCM simples, score enregistré

- Recommandations (MVP basique): contenus populaires + même matière; (V2.0) IA personnalisée

4.6 Admin (Back-office)

- Gestion de contenus: créer/éditer/supprimer départements, matières, cours, leçons

- Upload: via URLs présignées S3/Cloudinary; association aux leçons

- Gestion utilisateurs: liste, recherche, suspension (si abus)

- Gestion abonnements: vue des plans, abonnés actifs, transactions

- Analytics: vues par cours/leçon, watch time agrégé, taux de conversion

- Modération: commentaires (retirer, bannir, mots interdits basiques)

4.7 Internationalisation

- Langue par défaut: Français

- Arabe: traduction statique ultérieure (fichiers JSON via ngx-translate ou équivalent)

5) Parcours utilisateurs (User flows)

- Inscription → Connexion → Choix plan → Paiement Bankily/Masrivi/Sedad → Redirection succès → Accès premium → Consommation des contenus → Commentaires

- Navigation: Accueil → Recherche/Filtre → Page cours → Leçon (vidéo/PDF) → Marquer “vu” → Recommandations

- Mot de passe oublié: demande → email sécurisé → réinitialisation

- Admin: Connexion → Dashboard → Upload contenus (vidéo/PDF) → Publier → Suivi analytics

6) Spécifications techniques

6.1 Frontend (Angular + TailwindCSS / Angular Material)

- Version cible: Angular 17+ (standalone components), TypeScript strict

- UI: TailwindCSS + Angular Material pour composants accessibles

- Modules/Domaines (standalone features):

• core (services, interceptors, guards)

• shared (UI, pipes)

• auth (inscription/connexion)

• catalog (recherche, cours, listes)

• lesson (player vidéo, PDF viewer, commentaires)

• dashboard (tableau de bord étudiant)

• subscription (checkout, état)

• admin (CRUD contenus, analytics)

- Routage & Guards: AuthGuard, RoleGuard (Admin), SubscriptionGuard (premium)

- State management: NgRx Store + Effects (recherches, cache, abonnement)

- Internationalisation: ngx-translate (FR par défaut, AR statique ultérieure)

- Intégrations:

• Vimeo Player SDK / YouTube embed

• ngx-extended-pdf-viewer pour PDF

• GA4 (consentement) + Matomo (optionnel)

- Accessibilité: focus visibles, contrastes, clavier

- SEO: Angular Universal (SSR) pour pages publiques cours/catalogue (V1.0)

- Protection contenu (UI): superposition filigrane dynamique; désactivation clic droit; PiP/AirPlay désactivés; bouton “Télécharger” absent; masquage des URLs sources; bannière “captures interdites” sur contenus premium.

6.2 Backend (Node.js + Express.js)

- Architecture: monolithique modulaire (src/modules):

• auth, users, courses, lessons, assets, comments, subscriptions, payments, admin, search

- Authentification: JWT (access + refresh en cookies HttpOnly), bcrypt pour mots de passe

- Validation: Zod ou Joi sur les payloads

- Sécurité: Helmet, CORS restrictif, rate limiting, logs Pino

- Stockage fichiers: S3 ou Cloudinary via URLs présignées générées par l’API

- Vidéo: Vimeo (domain privacy + contrôle côté serveur), YouTube (non répertorié)

- Paiements: Bankily/Masrivi/Sedad REST (création d’ordre, capture, webhooks), états robustes

- Lecture vidéo/pdfs sécurisée: URLs signées courtes durées, clés tournantes (HLS AES-128 si applicable), vérification d’abonnement côté serveur, token de lecture par utilisateur/ressource, invalidation en cas de partage.

- Cache: Redis (optionnel) pour résultats de recherche populaires

- Observabilité: Sentry (erreurs), Healthcheck /healthz

6.3 Base de données (PostgreSQL)

- Tables principales:

• users(id, email unique, password\_hash, name, department\_id, semester, role, created\_at)

• departments(id, name)

• courses(id, department\_id, title, description, semester, professor, tags[], created\_at, is\_premium)

• lessons(id, course\_id, title, type[video|pdf|exam], duration\_sec, vimeo\_id|youtube\_id, pdf\_url, is\_premium, order\_index, created\_at)

• lesson\_assets(id, lesson\_id, kind[pdf|attachment], url, title)

• comments(id, lesson\_id, user\_id, content, status[visible|hidden|reported], created\_at)

• subscription\_plans(id, name, interval[month|year], price\_cents, currency)

• subscriptions(id, user\_id, plan\_id, status[active|canceled|expired], start\_at, end\_at, cancel\_at)

• payments(id, user\_id, subscription\_id, provider[bankily|masrivi|sedad], provider\_ref, amount\_cents, currency, status, created\_at)

• progress(id, user\_id, lesson\_id, status[viewed|in\_progress], updated\_at)

• coupons(id, code unique, discount\_percent, valid\_from, valid\_to, max\_redemptions) (V1.0)

• quizzes(id, lesson\_id, title) & quiz\_questions & quiz\_answers (V1.0)

- Indexes:

• users(email), lessons(course\_id, order\_index), comments(lesson\_id, created\_at), subscriptions(user\_id, status), payments(provider\_ref)

- Contraintes:

• FK avec cascade appropriée, checks sur types/énums, unicité sur codes/courriels

6.4 API (Contrats principaux)

- Auth:

• POST /api/auth/register {email, password, name, departmentId, semester}

• POST /api/auth/login {email, password}

• POST /api/auth/refresh

• POST /api/auth/forgot-password {email}

• POST /api/auth/reset-password {token, newPassword}

- Utilisateur:

• GET /api/me → profil + état abonnement

• PATCH /api/me {name, departmentId, semester}

- Catalogue & Cours:

• GET /api/catalog?query=&departmentId=&semester=&sort=popular|new|exam

• GET /api/courses/:id

• GET /api/courses/:id/lessons

• GET /api/lessons/:id (si premium et abonné, inclut sources protégées)

- Commentaires:

• GET /api/lessons/:id/comments

• POST /api/lessons/:id/comments {content}

• DELETE /api/comments/:id (auteur ou admin)

- Abonnements & Paiements (Bankily/Masrivi/Sedad):

• GET /api/plans

• POST /api/checkout {planId, couponCode?} → renvoie approvalUrl/intent

• POST /api/payments/webhook (webhook provider)

• GET /api/subscription → état actuel

- Admin:

• CRUD departments, courses, lessons, assets

• POST /api/admin/uploads/presign {type: pdf|asset} → {url, fields}

• GET /api/admin/analytics → vues, watch time, top cours

- Erreurs: format JSON {error: {code, message}} cohérent

6.5 Sécurité & conformité

- Hashage mots de passe: bcrypt (cost adapté)

- Tokens: JWT signés; refresh en HttpOnly, SameSite=Lax, rotation

- CORS: frontend origin(s) autorisés seulement

- Rate limiting: auth, commentaires, recherche

- Validation entrée/sortie, logs d’audit admin, RGPD (minimisation des données)

- Protection contre capture d’écran et enregistrement:

• Politique: interdiction stricte dans les CGU de toute capture (image/vidéo) des contenus premium; sanctions jusqu’à suspension et résiliation.

• Mesures techniques: filigrane dynamique sur player et PDF (ID utilisateur, email haché, horodatage, empreinte session); URLs signées à durée courte; HLS chiffré (AES-128) si applicable; privacy/domain-level sur Vimeo; désactivation du téléchargement, PiP et AirPlay; entêtes CSP/Referrer-Policy stricts; détection heuristique de screen recorders (quand possible) et journalisation.

• Traçabilité & réponse: logs corrélés (lecture, IP, device), révocation tokens en cas d’abus, mécanisme de signalement et enquête interne.

6.6 Performance, SEO, PWA

- Performance: lazy-loading routes, optimisation images, cache HTTP pour assets

- SEO (V1.0): SSR pour pages publiques, métadonnées OpenGraph

- PWA: manifest, offline basique pour pages shell (optionnel post-MVP)

6.7 Déploiement & environnements

- Environnements: dev, staging, prod

- Hébergement recommandé:

• Frontend: Vercel (Angular SSR supporté via adapteurs) ou Render static

• Backend: Render (Node/Express) ou AWS EC2

• DB: PostgreSQL managé (Neon/Render Postgres)

• Stockage: AWS S3 (ou Cloudinary pour PDFs/assets)

• Vidéo: Vimeo (privé; domaines autorisés); YouTube non répertoriée en fallback

- CI/CD: GitHub Actions (lint, tests, build, déploiement)

- Variables d’environnement (exemples):

• BACKEND: PORT, NODE\_ENV, DATABASE\_URL, JWT\_SECRET, JWT\_REFRESH\_SECRET, CORS\_ORIGINS, S3\_BUCKET, S3\_ACCESS\_KEY, S3\_SECRET\_KEY, VIMEO\_TOKEN, CLOUDINARY\_URL

• FRONTEND: API\_BASE\_URL, GA\_MEASUREMENT\_ID, VIMEO\_PLAYER\_KEY

7) Design & branding

- Couleurs:

• Bleu #1E3A8A (principal)

• Orange #F97316 (accent)

• Gris clair #F3F4F6 (fond)

• Gris foncé #111827 (texte)

• Vert #10B981 (succès)

- Typographies:

• Latin: Poppins / Montserrat / Inter

• Arabe: Cairo / Changa / Noto Kufi Arabic

- Iconographie: style outline (Heroicons/Material Icons)

- Composants clés:

• Header/Nav, Hero, Carte Cours, Liste Leçons, Player, PDF Viewer, Commentaires, Filtres, Pagination, CTA Abonnement, Tableau de bord cartes KPI

- Principes UX: lisibilité, hiérarchie claire, actions évidentes, mobile-first

8) Critères d’acceptation (extrait)

- Auth:

• Inscription/connexion fonctionnelles, erreurs claires, cookies sécurisés

• Réinitialisation mot de passe aboutit à un login réussi

- Catalogue:

• Recherche renvoie des résultats pertinents en < 500 ms (moyenne)

• Filtres combinables; tri cohérent

- Leçon:

• Player vidéo charge et joue sur desktop et mobile

• Filigrane dynamique visible en lecture, PiP/AirPlay désactivés, clic droit désactivé

• URLs sources non exposées; lecture refusée sans jeton valide

• PDF s’affiche correctement, téléchargement contrôlé si requis, filigrane appliqué

- Abonnement:

• Paiement Bankily/Masrivi/Sedad crée un abonnement actif après webhook

• Contenu premium inaccessible sans abonnement actif

- Admin:

• Upload via présign génère une ressource accessible et liée à la leçon

• Analytics affichent top 5 cours par vues

Fin du document.