NovaEdge Academy — System & Product Flow

Below is a clear, actionable flow for the entire product: user journeys, certificate issuance, system architecture, data model, key APIs, deployment/subdomain steps, MVP roadmap and next actions. Use this as the single source-of-truth for building the platform.

1) High-level user journeys (3 main personas)

Visitor

- Arrives on `novaedgeacademy.novaedgedigitallabs.tech` → views courses.
- Views course detail → can preview free lessons.
- Clicks Enroll → if paid redirect to Razorpay / simulated payment.
- After success → account created/linked → redirected to Dashboard.

Student

- Dashboard shows enrolled courses, progress, mentor, batch.
- Works through modules → marks lessons complete.
- Submits assignments / project links.
- On 100% completion → receives certificate (PDF + CertID + QR), optional LOR.
- Views / shares certificate; employer verifies at `/verify/[certId]`.

Admin / Instructor

- Login to Admin Dashboard → manage courses, batches, instructors.
- Approve project submissions; mark or verify completion.
- Trigger certificate re-issue or revoke.
- Monitor analytics and payments.

2) Certificate issuance & verification flow (detailed)

- Student completes last lesson → frontend sends `POST /api/progress/complete-lesson`.
- Backend marks lesson complete & recalculates course progress.
- If progress === 100:
  - Create certificate metadata record in DB with status `pending`.
  - Enqueue a job (Redis / simple background worker).
- Worker job:
  - Renders certificate HTML via Puppeteer (headless Chromium).
  - Generates QR linking to `/verify/[certId]`.
  - Uploads resulting PDF to S3/Cloudinary.
  - Saves `{ certId, studentId, courseId, pdfUrl, issuedAt, signature, status: "active" }` to MongoDB.
  - Sends email + notification to student with PDF link.
- Public verification:
  - Employer opens `/verify/[certId]` → API fetches certificate record → shows student name, course, issuedAt, pdf thumbnail, verification status.
- Admin UI can revoke → sets status to `revoked` and optionally logs reason.

Security & tamper-resistance:

- Cert metadata stored immutable (append log).
- Optionally store certificate hash (SHA256) in DB for integrity; optionally anchor hash to a public timestamping service later.

3) Core system architecture (components)

- **Frontend:** Next.js (app router) — React, Tailwind, Radix UI
- **Backend:** Node.js + Express (API server) — or Next.js serverless API routes for smaller scale
- **DB:** MongoDB Atlas
- **File Storage:** AWS S3 or Cloudinary (certs, uploads)
- **Queue/Worker:** BullMQ/Redis or simple node child process (for Puppeteer jobs)
- **Payments:** Razorpay (live) + Simulated payments (dev)
- **Auth:** JWT access + refresh stored in HttpOnly cookies
- **Hosting:** Vercel for frontend, Render/Heroku/EC2 for backend if not using serverless
- **Monitoring:** Sentry (errors), Google Analytics/GA4 for web metrics, MongoDB charts or Metabase for business analytics

Diagram (linear):

```
User (browser)
  :
Next.js Frontend (Vercel)
  :
API Server (serverless or Express)
  :          :
MongoDB Atlas   S3/Cloudinary
  :
Background Worker (Puppeteer) → upload PDF → update DB → notify user
```

4) Minimal DB schema (MongoDB collections — core fields)

users

```
_id, name, email, passwordHash, role["student","admin","instructor"], createdAt, p
```

courses

```
_id, title, slug, price, description, modules:[{id,title,lessons:[{id,title,type,d
```

enrollments

```
_id, userId, courseId, paymentId, status["active","cancelled"], enrolledAt, progre
```

progress

```
_id, userId, courseId, moduleId, lessonId, completedAt
```

projects/submissions

```
_id, userId, courseId, submissionUrl, score, mentorComments, verified (bool), subm
```

certificates

```
_id, certId, userId, courseId, pdfUrl, issuedAt, issuedBy, status["active","revoke
```

payments

```
_id, userId, courseId, providerId, amount, status, providerResponse, createdAt
```

auditLogs

```
_id, action, actorId, targetId, details, timestamp
```

5) Key API endpoints (suggested)

Auth

- `POST /api/auth/signup`
- `POST /api/auth/login` → set HttpOnly access cookie + refresh cookie
- `POST /api/auth/refresh`
- `POST /api/auth/logout`

Courses

- `GET /api/courses` (list)
- `GET /api/courses/:slug` (detail)
- `POST /api/admin/courses` (admin)
- `PUT /api/admin/courses/:id` (admin)

Enrollment & Payments

- `POST /api/enroll` → create enrolment + return Razorpay order or simulated success
- `POST /api/payments/webhook` → verify & update enrollment

Progress & Submissions

- `POST /api/progress/complete-lesson` → check progress & trigger certificate
- `POST /api/submissions` → upload project link
- `POST /api/admin/verify-submission` → approve project

Certificates

- `GET /api/certificates/:certId` → public verification
- `GET /api/user/certificates` → private user list
- `POST /api/admin/certificates/revoke` → admin revoke

Admin

- `GET /api/admin/analytics` (enrollments, completions)
- `POST /api/admin/instructors` etc.

6) Security & infra checklist

- Use HTTPS everywhere, HSTS.
- Keep JWT in HttpOnly, Secure cookies. Use short-lived access tokens + refresh rotation.
- Rate limit critical endpoints (login, enroll).
- Sanitize uploads and limit file sizes.
- Use strong CSP for frontend pages that render certificate previews.
- Ensure Puppeteer runs in a secure environment (sandbox, image processing limits).
- Backups for MongoDB and S3.

7) Subdomain & deployment steps (quick)

1. Create subdomain: `novaedgeacademy.novaedgedigitallabs.tech` in your DNS (A/ CNAME to Vercel).
2. Add project in Vercel for frontend; set branch & build config (NEXT_PUBLIC_API_URL, etc.).
3. For backend, deploy to Render/Vercel serverless; set env variables (MONGO_URI, S3 keys, RAZORPAY keys).
4. Configure CORS to allow `novaedgeacademy` origin.
5. Add SSL (auto on Vercel/Render).
6. Test flows end-to-end on staging before DNS switch.

8) MVP roadmap (6–10 weeks, lean)

Week 1 — Core infra & auth

- Project skeleton (Next.js + Express or Next API).
- Auth with JWT cookies.
- Basic UI: homepage, course listing, course detail.

Week 2 — Course & Enrollment

- Course CMS (admin create).
- Enroll + simulated payment flow.
- Student dashboard (enrolled list, progress bar).

Week 3 — Lessons & Progress

- Lessons UI, mark complete API.
- Progress calculation.

Week 4 — Certificates

- Puppeteer worker flow to generate PDF.
- Store PDF to S3/Cloudinary.
- Public verify page `/verify/[certId]`.

Week 5 — Admin UI & analytics

- Admin dashboard to approve projects & revoke certs.
- Basic analytics.

Week 6 — Polishing & payments

- Razorpay integration.
- Email notifications, LOR generation.
- Deploy to production, link subdomain.

Optional Weeks: Job placements, cohort features, payment reconciliation, mobile optimizations.

9) Metrics to track (KPIs)

- Enrollments per week
- Completion rate per course
- Certificate issuance count
- Active users (DAU/MAU)
- Conversion rate (visit → enroll)
- Revenue / refunds
- Job placements or internships placed (later)