**High-level architecture (Replit-ready)**

**Monorepo layout**

* /client — React + Vite + Tailwind + shadcn/ui for the web app (SPA).
* /server — Node/Express (or Fastify) API with Drizzle ORM over Postgres (or MySQL).
* /shared — Zod schemas, DTOs, and type contracts shared between client and server.
* /attached\_assets — static templates (e.g., NPA certificate HTML, logos).
* Infra: .replit and replit.md for Replit run commands & environment; GitHub Actions for CI. ([GitHub](https://github.com/jitubagria/Yahle))

**Core tech**

* Frontend: React + Vite + Tailwind + shadcn/ui, React Router, TanStack Query.
* Backend: Node 22 + Fastify/Express, Drizzle ORM, Zod validation, JWT auth, Multer/S3 for uploads.
* DB: Postgres (preferred) or MySQL (compatible with your existing stacks).
* Search: PostgreSQL full-text or Meilisearch (optional).
* Realtime (for quizzes, dashboards): Socket.IO or Server-Sent Events.
* Jobs/Cron: node-cron (Replit always-on), or Better: a lightweight queue (BullMQ + Redis) if needed.
* PDFs: Puppeteer (Chromium) for server-side PDF generation (NPA certs, profiles, invoices).
* Emails/SMS: Resend/SMTP + SMS (MSG91/Twilio); WhatsApp share links on client.
* Payments: Razorpay/PhonePe (for paid quizzes/courses/masterclasses).
* Observability: pino logs + Health endpoints + simple metrics.

**Modules & core functions**

Below are the **modules** you asked for across chats (Doctors, Hospitals, Jobs, Courses, Quizzes, Masterclass, AI Tools, Research, Support, Causes/“Medical Voices”, Home-Care & Telemedicine, Conferences). For each, I list **core functions** (API + UI), minimal DB, and notes.

**1) Auth & Accounts (foundation)**

**Core functions**

* OTP Login (SMS) and Email+Password; JWT access+refresh; roles: admin/moderator/user.
* Profile basics (name, gender, phone/email verification, KYC optional).
* Org accounts for hospitals/clinics (multi-user)
* Notifications (in-app + email/SMS).

**DB (tables)**

* users, user\_sessions, user\_roles, otp\_verifications, notifications.

**2) Doctors Directory**

**Core functions**

* Create doctor profile (UG/PG/SS/Additional), job details, clinic/hospital affiliations.
* Search & filters (specialty, city, hospital, qualification, branch/year).
* Profile page PDF export (for “print/share”).
* Favorites / Save / Share (WA link).
* Admin bulk import (CSV/Excel).

**DB**

* doctors, doctor\_education (UG/PG/SS/addl), doctor\_jobs, doctor\_affiliations, favorites.

**API**

* GET /api/doctors?specialty=&city=&...
* POST /api/doctors (admin/self), GET /api/doctors/:id, PATCH /api/doctors/:id
* POST /api/favorites {doctor\_id} / GET /api/favorites (you’ve asked this for Heart-Foundry too)

**3) Hospitals Directory**

**Core functions**

* Add hospital/clinic, departments, services, facilities.
* Search by city/type/department; map view later.
* Link doctors to hospitals.

**DB**

* hospitals, hospital\_departments, hospital\_services, doctor\_hospital\_links.

**API**

* GET /api/hospitals?..., POST /api/hospitals, GET /api/hospitals/:id, PATCH ...

**4) Jobs (custom WP-style plugin, but native here)**

**Core functions**

* Post a job & Find jobs on **same page** (filters: specialty, city, experience, salary).
* Employer dashboard (manage postings, applicants).
* Candidate apply (attach profile/resume), shortlist, status updates.

**DB**

* jobs, job\_applications, employers, employer\_memberships (for paid tiers).

**API**

* GET /api/jobs?..., POST /api/jobs, POST /api/jobs/:id/apply, GET /api/applications?mine

**5) Courses (LearnPress-like)**

**Core functions**

* Course catalog, detail (overview, syllabus, faculty, FAQs, reviews).
* Enrollment (free/paid), lessons (video, docs), simple quizzes, certificates of completion.
* Separate **BCBR** course track & **Masterclasses** integration (see below).

**DB**

* courses, course\_lessons, course\_enrollments, course\_reviews, payments.

**API**

* GET /api/courses?..., GET /api/courses/:id, POST /api/courses/:id/enroll, GET /api/my/courses

**6) Quizzes (standalone + within courses)**

**Core functions**

* **Live quiz mode**: synchronized questions, per-question leaderboard (correctness + speed).
* **On-demand mode**: user can practice anytime.
* Admin builder: MCQ with optional image; timer; scoring rules.
* Results dashboards, export CSV.

**Realtime**

* Socket room = quiz instance; push question → clients lock answers → compute ranks → push leaderboard.

**DB**

* quizzes, quiz\_questions, quiz\_instances, quiz\_responses, quiz\_scores.

**API**

* POST /api/quizzes (admin), POST /api/quizzes/:id/start (creates instance)
* WS: /ws/quiz/:instanceId for broadcast; POST /api/quizzes/:instanceId/answer

**7) Masterclass (offline workshops, NPNL)**

**Core functions**

* Catalog (Medicolegal core), events by city/date; registration (paid/free).
* Attendee list, check-in QR, completion certificate PDF.

**DB**

* masterclasses, masterclass\_sessions, registrations, certificates.

**API**

* GET /api/masterclasses, POST /api/masterclasses/:id/register, GET /api/my/masterclasses

**8) AI Tools**

**Initial set**

* **Automate NPA Certificate**: scheduled monthly PDF generation on a fixed day; cron sends email/WhatsApp.
  + Inputs: doctor details, period, designation, place; pre-designed A4 HTML template.
  + Admin dashboard to monitor sent/failed.
* **Sample Size Calculator**, **Stats Explainer**, **Research Tools** (common UI shell).

**DB**

* ai\_tools\_users, npa\_profiles, npa\_runs (status/sent\_at/log), ai\_tool\_usage.

**API**

* POST /api/ai/npa/setup, POST /api/ai/npa/preview, POST /api/ai/npa/run (manual), (Cron auto)
* GET /api/ai/usage?tool=npa

**Jobs**

* cron: 0 8 25 \* \* (example) → generate PDFs via Puppeteer → email & WhatsApp link → log status.

**9) Research**

**Core functions**

* Article writing support (structured sections), templates (RUHS style), reference management.
* Upload data (Excel) → basic descriptive stats & charts → export DOCX/PDF slides.
* Publication tracker (journal selection, status).

**DB**

* research\_projects, research\_docs, datasets, analysis\_runs, publications.

**API**

* POST /api/research/projects, POST /api/research/:id/upload, POST /api/research/:id/analyze

**10) Medical Voices (Causes)**

**Core functions**

* Cause landing page (banner, description, linked docs/images, authority/department tags).
* Join/Support (opt-in), WhatsApp/Twitter share.
* Optional **gathering** details (date/time/location, contact leader; preset leaders per cause).
* Admin: create cause form with fields you listed.

**DB**

* causes, cause\_docs, cause\_supporters, cause\_contacts, cause\_events.

**API**

* GET /api/causes?..., POST /api/causes, POST /api/causes/:id/support, GET /api/causes/:id/events

**11) Home-based Care & Telemedicine (combo)**

**Core functions**

* Request home visit/teleconsult; service catalog; geo-filtering by city/area.
* Slot booking; payments; visit notes, receipts.

**DB**

* homecare\_services, telemedicine\_slots, appointments, visit\_notes, invoices.

**API**

* POST /api/homecare/requests, GET /api/telemedicine/slots, POST /api/appointments

**12) Conferences**

**Core functions**

* List conferences (filters by specialty/location/date); add session agenda; registration links.
* Export to calendar (.ics).

**DB**

* conferences, conference\_sessions, conference\_registrations.

**API**

* GET /api/conferences?..., POST /api/conferences, GET /api/conferences/:id

**13) Support/Helpdesk**

**Core functions**

* Ticketing (categories: login, payments, course, quiz, AI tools).
* Canned responses; SLA; status; file attachments.

**DB**

* tickets, ticket\_comments, ticket\_attachments.

**API**

* POST /api/support/tickets, GET /api/support/tickets?mine, POST /api/support/tickets/:id/reply

**Database map (condensed)**

* **Auth**: users, user\_sessions, otp\_verifications, user\_roles, notifications
* **Directory**: doctors, doctor\_education, doctor\_jobs, hospitals, doctor\_hospital\_links, favorites
* **Content**: courses, course\_lessons, course\_enrollments, course\_reviews
* **Quizzes**: quizzes, quiz\_questions, quiz\_instances, quiz\_responses, quiz\_scores
* **Masterclass**: masterclasses, masterclass\_sessions, registrations, certificates
* **AI Tools**: ai\_tool\_usage, npa\_profiles, npa\_runs
* **Research**: research\_projects, research\_docs, datasets, analysis\_runs, publications
* **Jobs**: jobs, job\_applications, employers, employer\_memberships
* **Voices**: causes, cause\_docs, cause\_supporters, cause\_contacts, cause\_events
* **Care/Telemed**: homecare\_services, telemedicine\_slots, appointments, visit\_notes, invoices
* **Conferences**: conferences, conference\_sessions, conference\_registrations
* **Support**: tickets, ticket\_comments, ticket\_attachments
* **Payments**: payments, refunds, coupons (shared)

Use **Drizzle** migrations (already present config in repo root) to keep schema in code. ([GitHub](https://github.com/jitubagria/Yahle))

**API design conventions**

* REST + WebSocket for realtime.
* **Auth**: Bearer access token; refresh token rotation.
* **Validation**: Zod on both client (shared/) and server.
* **Errors**: RFC7807-style JSON; consistent error codes.
* **Pagination**: ?page=&limit=&q=&sort=.
* **Uploads**: /api/uploads returns public URL (S3/Cloudflare R2) with signed PUTs.
* **Rate limits**: IP+user based (express-rate-limit).

**Frontend app shell (client)**

* **Top Nav** (your preferred): Home | Doctors | Hospitals | Jobs | Courses | Quizzes | AI Tools | Masterclass | Research | Medical Voices | Care | Conferences | Support | Contact
* **Utility**: Login/Signup, Dashboard, Search, Notifications, Cart
* **Pages**:
  + List/detail pages for each module, with filters & share buttons.
  + Dashboard with tabs: My Profile, Favorites, My Courses, My Quizzes, My Registrations, My Jobs, My AI Tools, Support Tickets, Billing.

**Background jobs & scheduling**

* **NPA Automate** cron (fixed day/time) → generate PDF → email + WhatsApp deep-link → log in npa\_runs.
* **Quiz live sessions**: scheduler opens/locks questions, rolls leaderboards.
* **Reminders**: jobs for class start, conference sessions (email/SMS).
* **Data hygiene**: nightly cleanup for sessions, expired OTPs, stale carts.

**Security & compliance**

* Env secrets via Replit Secrets / .env (never commit).
* Input validation everywhere; file-type checks; PDF render sandbox.
* Least-privilege DB user; IP allowlist for admin; audit logs.
* GDPR-style data exports/deletes for users.

**Observability**

* /health (server), /metrics (basic counters).
* **pino** logs with request IDs; error aggregation.
* Admin screen for job statuses (NPA, emails, quiz instances).

**CI/CD (GitHub → Replit)**

* **CI**:
  + Lint + Type-check + Build client/server
  + Unit tests (Jest) and minimal API smoke tests (Supertest).
* **CD**:
  + On main push, deploy to Replit (use Replit GitHub integration or Actions + Replit API).
* Keep Ports: client dev on 5173, server on 5000; production serves client via CDN and server on 8080 (Replit default).

Your repo already contains .replit / replit.md (great for wiring run commands) and Vite/Tailwind/Drizzle configs — we’ll hook workflows to that. ([GitHub](https://github.com/jitubagria/Yahle))

**Milestone plan (you can follow this order)**

**M1 — Foundations**

* Set DB (Postgres on Replit or Neon.io).
* Implement Auth (OTP + email), /health, logging.
* Build shared Zod schemas (shared/).

**M2 — Directories**

* Doctors + Hospitals: CRUD, filters, profile PDF, favorites.
* Import tool (CSV).

**M3 — Jobs**

* Post & find on one page, applications, employer dashboard.

**M4 — Courses & Masterclass**

* Course catalog & enrollments; masterclass events & ticketing; certificate PDF.

**M5 — Quizzes**

* Admin quiz builder; **live mode** (Socket.IO); leaderboards; exports.

**M6 — AI Tools (NPA first)**

* NPA setup → preview → cron send; usage analytics.

**M7 — Research**

* Project templates; basic stats; export DOCX/PPT.

**M8 — Voices, Care, Conferences, Support**

* Causes join/share/gathering; home-care & telemedicine booking; conferences; ticketing.

**M9 — Polish**

* Payments, receipts, invoices; metrics; admin analytics.

If you want, I can generate:

* Drizzle schema starter files for each table group,
* A ready routes/ tree with controllers (Express/Fastify),
* React routes and component stubs (shadcn/ui),
* A GitHub Actions CI yaml tailored to Replit.

Just say “generate the scaffolding,” and I’ll drop the code structured exactly for your Yahle repo.