**user-management-system**

Below is a presentation‑ready, end‑to‑end guide you can hand to teammates, stakeholders, or use live. It covers architecture, local dev, VS Code, GitHub/CI, Docker/K8s/Helm, testing, monitoring, security, error handling, and a timed demo script.

**Executive Summary**

* Full‑stack UMS with auth, profiles, admin listing, and account deletion.
* Dual DB: PostgreSQL (auth/audit) + MongoDB (profiles/preferences).
* JWT auth, Zod validation, structured error payloads, Swagger docs.
* React + TypeScript frontend with Redux Toolkit and RHF + Zod forms.
* DevOps: Docker, Kubernetes, Helm (dev/prod), Terraform (AWS), CI via GitHub Actions.
* Monitoring: Health endpoint and Prometheus metrics. Realtime: Socket.IO heartbeat.

**Repository Map**

* Backend: Express app, routes, services, middlewares, DB connectors, tests
  + backend/src/app.js:1 (app composition), backend/src/routes/\*.js, backend/src/services/\*.js
  + backend/prisma/schema.prisma:1 (Postgres schema)
  + backend/src/models/userProfile.model.js:1 (Mongo schema)
  + backend/tests/api.test.js:1 (REST tests)
* Frontend: React/TS SPA, pages/components/store/tests
  + frontend/src/App.tsx:1, frontend/src/pages/\*.tsx, frontend/src/components/\*.tsx
  + frontend/src/store/slices/authSlice.ts:1 (Redux)
  + frontend/src/pages/\_\_tests\_\_/Login.test.tsx:1 (component test)
* Infra: Docker, K8s, Helm, Terraform
  + infrastructure/docker-compose.yml:1, infrastructure/kubernetes/\*.yaml
  + infrastructure/helm/ums/\*\* (Helm chart)
  + infrastructure/terraform/\*.tf (AWS S3/ECR)
* CI: .github/workflows/ci.yml:1
* Docs: docs/api/openapi.md:1, Swagger served at /docs

**Architecture Overview**

* API: Express, layered by concerns
  + Routes: backend/src/routes/auth.routes.js:1, backend/src/routes/user.routes.js:1
  + Services: backend/src/services/auth.service.js:1, backend/src/services/profile.service.js:1
  + Middlewares: backend/src/middlewares/auth.js:1 (JWT/admin guard), backend/src/middlewares/errors.js:1 (structured errors)
  + Config: backend/src/config/env.js:1 (env wiring)
* Databases
  + PostgreSQL (Prisma): UserAuth + AuditLog (backend/prisma/schema.prisma:15)
  + MongoDB (Mongoose): UserProfile with email index and preferences (backend/src/models/userProfile.model.js:1)
* Realtime: Socket.IO heartbeat (backend/src/server.js:1)
* Observability: /health and /metrics (backend/src/api/health.js:1)

**Security & Validation**

* JWT signed by JWT\_SECRET (backend/src/lib/jwt.js:1)
* Zod validation with aggregated issues (backend/src/lib/validate.js:1)
* Helmet + CORS + rate limiting (backend/src/app.js:1)

**Error Handling Model**

* Server responses include message, name, reason, status, details, stack for exceptions
  + backend/src/middlewares/errors.js:1
* Frontend surfaces full details in StatusBar (renders as preformatted text)
  + frontend/src/components/StatusBar.tsx:1
* Error formatting covers Axios and generic errors
  + frontend/src/util/errors.ts:1

**Capabilities & Endpoints**

* Register: POST /auth/register (user/admin with secret), JWT returned
* Login: POST /auth/login, JWT returned
* Me: GET /me returns merged profile/role; PUT /me updates profile and email; DELETE /me deletes account across Mongo + Postgres
* Admin: GET /admin/users lists all profiles (ADMIN only)
* Docs: /docs (Swagger), Health: /health, Metrics: /metrics

**Local Development (Terminals)**

* Prereq: Docker Desktop
* Quick start (compose, migrations, URLs):
  + powershell -ExecutionPolicy Bypass -File .\run.ps1 up
  + URLs: API http://localhost:4000, Swagger http://localhost:4000/docs, Frontend http://localhost:5173
* Helper commands:
  + .\run.ps1 restart | down | logs | ps | seed
* Without Docker:
  + Start Postgres+Mongo locally
  + Backend: cd backend && npm i && npx prisma generate && npm run dev
  + Frontend: cd frontend && npm i && npm run dev

**VS Code Dev Flow**

* Open folder, enable TS/ESLint suggestions
* Split terminals: one for backend (npm run dev), one for frontend (npm run dev)
* Set breakpoints in routes/services; restart Nodemon automatically
* Run tests:
  + Backend: cd backend && npm test -- --runInBand --verbose
  + Frontend watch: cd frontend && npm run test:watch
  + Frontend CI: cd frontend && npm run test:ci

**GitHub & CI/CD**

* CI Workflow .github/workflows/ci.yml:1
  + Backend job: spins Postgres/Mongo services, runs Prisma generate/migrate deploy with injected env, executes Jest tests
  + Frontend job: installs deps and builds Vite app
* Recommended next step: add a release workflow to build/push Docker images to GHCR and optional Helm deploy to a cluster via GitHub Environments

**Docker Images**

* Backend backend/Dockerfile:1 (Node 20/Alpine, prisma generate, node src/server.js)
* Frontend frontend/Dockerfile:1 (Node build → Nginx serve)

**Kubernetes & Helm**

* Manual K8s (manifests present) or standardized Helm:
  + Chart: infrastructure/helm/ums/
  + Dev deploy example:
    - Create namespace and secret:
      * kubectl create ns ums
      * kubectl -n ums create secret generic ums-secrets --from-literal=JWT\_SECRET=... --from-literal=ADMIN\_INVITE\_SECRET=... --from-literal=DATABASE\_URL=... --from-literal=MONGODB\_URI=...
    - Deploy:
      * helm upgrade --install ums infrastructure/helm/ums -f infrastructure/helm/ums/values.yaml -f infrastructure/helm/ums/values-dev.yaml
  + Prod: use values-prod.yaml (replicas/resources, CORS)

**Terraform (AWS)**

* Scaffolds S3 and ECR repos
  + cd infrastructure/terraform && terraform init && terraform apply
* Set AWS credentials or use OpenID Connect in GitHub Actions for push

**Testing Strategy & Commands**

* Backend (Jest + Supertest): full REST suite with request+response logging (backend/tests/api.test.js:1)
  + Run: cd backend && npm test -- --runInBand --verbose
* Frontend (Vitest + RTL):
  + Unit/Component: cd frontend && npm run test:ci or npm run test:watch
  + Current specs cover reducer lifecycles, StatusBar rendering, login page presence

**Monitoring & Troubleshooting**

* Health check: GET /health returns { status: "ok" }
* Metrics: GET /metrics returns Prometheus stats (set ENABLE\_METRICS=true)
* Common issues:
  + Prisma env missing: set DATABASE\_URL where Prisma runs (handled in CI)
  + Admin registration disabled: set ADMIN\_INVITE\_SECRET
  + 401s: ensure Authorization: Bearer <token> header set, JWT secret matches

**Security Posture**

* Helmet, CORS (per env), rate limiting (15m window)
* JWT tokens with role claims; admin guard on admin endpoints
* Input validation via Zod; error payloads include details for client UX without leaking secrets

**Error Handling UX**

* Backend error middleware returns rich payloads (backend/src/middlewares/errors.js:1)
* Frontend StatusBar shows full message/reason/details/stack (for dev) and renders as <pre> for readability (frontend/src/components/StatusBar.tsx:1)

**15‑Minute Demo Script**

* 0:00–1:00 Project overview (README), repo structure, architecture diagram (verbally)
* 1:00–3:00 Local start
  + .\run.ps1 up → show Swagger /docs and frontend
* 3:00–6:00 Auth flows
  + Register USER, auto fetch /me; show token in devtools, login, profile load
* 6:00–9:00 Profile update + delete
  + Update name/email/preferences; show persistence in /me; delete account and verify 404/401
* 9:00–11:00 Admin flow + realtime
  + Register ADMIN with secret; visit Admin page; demonstrate heartbeat messages
* 11:00–12:30 Monitoring
  + /health, /metrics output; note rate limiting header if applicable
* 12:30–13:30 Tests
  + Backend: npm test -- --runInBand --verbose
  + Frontend: npm run test:ci
* 13:30–14:30 CI workflow
  + Show Actions run, database services, Prisma, tests/build
* 14:30–15:00 K8s/Helm
  + Show chart values and sample helm upgrade --install

**Operational Checklists**

* Local
  + Docker Desktop running
  + .\run.ps1 up succeeds; /docs and frontend reachable
  + Seed users if needed .\run.ps1 seed
* CI
  + Actions green for backend + frontend jobs
  + Prisma migrations ran in CI
* K8s deploy
  + Namespace ums exists
  + Secret ums-secrets created with envs
  + Helm chart installed with appropriate values
  + Services respond at ingress/load balancer
* Security
  + CORS\_ORIGIN correct for environment
  + Strong JWT\_SECRET, non‑default admin secret
* Monitoring
  + /metrics scraped by Prometheus
  + Logs/alerts configured

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a login screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Below are precise, presentation‑ready, step‑by‑step UI flows that exercise every feature. Use them as a live demo script or QA checklist.

**Prerequisites**

* Start stack: powershell -ExecutionPolicy Bypass -File .\run.ps1 up
* Frontend: [http://localhost:5173](http://localhost:5173/)
* Admin secret: the value in backend ADMIN\_INVITE\_SECRET (compose/README defaults to let-admins-in)

**UI Basics**

* Navigation: Home, Profile, Admin (Admin link only appears for admins once signed in)
* StatusBar: fixed at bottom; shows current status and full error text in a preformatted block

**Flow 1 — Register Standard User**

* Open [http://localhost:5173](http://localhost:5173/)
* Click “Register”
* Fill:
  + Name: Test User
  + Email: [user@test.dev](mailto:user@test.dev)
  + Password: Password1!
  + Role: leave as Standard user
* Click “Create account”
* Expect:
  + StatusBar: “Account created and profile loaded”
  + Redirect to Profile with populated name/email/preferences
  + Token stored in localStorage

**Flow 2 — Login Existing User**

* Click “Login”
* Fill:
  + Email: [user@test.dev](mailto:user@test.dev)
  + Password: Password1!
* Click “Login”
* Expect:
  + StatusBar: “Logged in and profile fetched”
  + Redirect to Profile
  + Admin link is hidden (non‑admin)

**Flow 3 — View Profile**

* Navigate to “Profile”
* Expect:
  + Name: Test User
  + Email: [user@test.dev](mailto:user@test.dev)
  + Theme: Light
  + Language: en
  + StatusBar: “Profile loaded” after the initial fetch

**Flow 4 — Update Profile (name, email, preferences)**

* On Profile:
  + Update Name: Updated User
  + Update Email: [updated@test.dev](mailto:updated@test.dev)
  + Change Theme: Dark
  + Change Language: es
* Click “Save changes”
* Expect:
  + StatusBar during save: “Saving profile…”
  + StatusBar success: “Profile updated”
  + Fields persist with new values
  + Subsequent “Login” works with new email ([updated@test.dev](mailto:updated@test.dev), same password)

**Flow 5 — Validation Errors (forms show full messages)**

* Register page:
  + Enter invalid email (user@@test)
  + Tab out or submit
  + Expect inline error label: “string: Invalid email” (shows type + message)
* Profile page:
  + Set Email = not-an-email
  + Click “Save changes”
  + Expect StatusBar error block to include:
    - message with field and code (invalid\_string)
    - name: ValidationError
    - status: 400
    - details array with path and message

**Flow 6 — Delete Account**

* On Profile, click “Delete account”
* Expect:
  + StatusBar during: “Deleting account…”
  + StatusBar success: “Account deleted”
  + Redirect to Home (guest session)
  + Attempt “Login” using deleted email should fail (see Flow 10)

**Flow 7 — Register Admin**

* Click “Register”
* Fill:
  + Name: Admin
  + Email: [admin@test.dev](mailto:admin@test.dev)
  + Password: Password1!
  + Role: Administrator
  + Admin secret: enter the backend secret (e.g., let-admins-in)
* Click “Create account”
* Expect:
  + StatusBar: “Account created and profile loaded”
  + Admin link now visible in the nav (conditional on role)

**Flow 8 — List Users (Admin)**

* Click “Admin”
* Expect:
  + Table with columns: Name, Email, Role, Theme, Language
  + At least one row for Admin and prior test users
  + StatusBar: “Loaded X users”

**Flow 9 — Realtime Heartbeat (Admin page)**

* Stay on Admin page for ~30 seconds
* Expect:
  + StatusBar updates: “Realtime heartbeat HH:MM:SS”
  + Confirms Socket.IO channel is active

**Flow 10 — Auth Failures (full error payload shown)**

* From Home (not logged in):
  + Try to open “Profile” directly at <http://localhost:5173/profile>
  + Expect redirect to Login; if any request fires, StatusBar error “TOKEN\_MISSING”
* Try login with incorrect password:
  + Email: [user@test.dev](mailto:user@test.dev)
  + Password: WrongPass!
  + Expect StatusBar error: includes reason INVALID\_CREDENTIALS
* Admin secret wrong:
  + Register admin with wrong secret
  + Expect StatusBar error: reason ADMIN\_SECRET\_INVALID

**Flow 11 — Role Guard**

* As a standard user (not admin):
  + Admin link should not be visible
  + If navigating to <http://localhost:5173/admin/users> directly:
    - Expect StatusBar error (admin required) and navigation back to Profile

**Flow 12 — Token Persistence + Auto Fetch**

* After a successful login/registration:
  + Refresh browser (F5)
  + Expect:
    - Protected routes reload user via GET /me using localStorage token
    - StatusBar briefly: “Loading profile…” then “Profile loaded”

**Flow 13 — Expired/Invalid Token Behavior (optional)**

* Manually set an invalid token in DevTools:
  + localStorage.setItem('token', 'invalid')
  + Visit Profile
  + Expect StatusBar error with reason TOKEN\_INVALID and details containing JWT error

**Flow 14 — Swagger & API Direct**

* Open <http://localhost:4000/docs>
* Try:
  + POST /auth/register (user), then GET /me with Bearer token
  + PUT /me to update preferences
  + GET /admin/users with admin token
* Compare responses with UI results to validate parity

**Flow 15 — Responsive Check**

* Narrow browser width below ~720px
* Expect:
  + Layout stacks, StatusBar expands to full width at bottom
  + Forms remain usable on mobile sizes

**Flow 16 — Logs & Metrics (for demos)**

* Backend logs (another terminal):
  + .\run.ps1 logs
  + Expect request logs (morgan) while performing flows
* Metrics:
  + curl <http://localhost:4000/metrics>
  + Show Prometheus output (e.g., process\_cpu\_user\_seconds\_total)

**Flow 17 — Seed Demo Data (optional)**

* .\run.ps1 seed
* Reload Admin page:
  + Expect seeded users present in the table