MedLearn overview

MedLearn is a training and onboarding LMS for pharmacies. It has public marketing pages and inquiry capture, plus an authenticated dashboard for approvals, invitations, user management, and course prototypes.

Tech stack

Frontend: Parcel bundler, vanilla JS modules, modular CSS partials.
Backend: Node.js + Express, mysql2, express-validator, cookie-session.
Database: MySQL.
Email: Mailgun for approvals and invites.

Project layout

src/

client/

public/

```
assets/

  css/
  styles.css (entry)
  settings.css (CSS variables)
  base.css (resets + base type)
  layout.css (header, grid, footer)
  components.css (buttons, cards, nav, forms)
  utilities.css (helpers and utility classes)
  vendor.css (vendor overrides)

assets/

  js/

    lib/
    main.js (shared nav + alerts + authFetch + bootstrapping)
    form-ui.js (shared helpers setSubmitBusy, checkSubmit, onSubmit)

    public/
    index.js
    inquiries.js
    onboarding.js
    signin.js
```

secure/

```
dashboard.html
admin/admin.html
```

server/

server.js (Express app + static serving)
routes/
inquiriesRoute.js
usersRoute.js
onboardingRoute.js
invitesRoute.js
companiesRoute.js

dist/ (Parcel output, gitignored)

Static serving and URLs

Express serves src/client/public at / so runtime asset paths are /assets/...
Examples:
/ → index.html
/inquiries.html → src/client/public/inquiries.html
/assets/... → src/client/public/assets/...

CSS organization

styles.css imports in this order: settings, base, layout, components, utilities, vendor.
Key utilities added: opacity-50, cursor-not-allowed, and a disabled style for .btn-primary to ensure gray appearance on first paint.

Role-aware UI

Admin pill in the navbar is hidden by default and revealed for admins after a session check.
main.js pattern: call authFetch on DOMContentLoaded, add body role class or unhide the Admin link.
Active-link highlighting uses pathname matching and a small special-case to light up Admin when on /secure/admin.

Session and authFetch

GET /api/session returns the shape stored in req.session.user at login.
Client uses authFetch() to read the session and gate UI.
Top-level awaits require scripts to be type="module"; otherwise wrap boot in DOMContentLoaded.

Onboarding flow (single page, role-aware)

Purpose: first-time setup for any invitee manager or employee. One page adapts to role using server-provided schema.

Entry points
Public URL: /onboarding.html?token=...
Pretty helper route optional: GET /onboard → 302 to /onboarding.html with original query string.

Frontend onboarding.js
Read token from URLSearchParams.
Call GET /api/onboarding?token=... to fetch context.
Prefill locked fields and editable fields.
Remove token from the URL with history.replaceState.
On submit, POST /api/onboarding/complete with FormData including token.
Use form-ui.js helpers for submit button state.

Locked vs editable
Locked display only but submit via hidden inputs when needed: company, role.
Email is readonly with a future Change email flow.
Editable: firstName, lastName, password, timeZone, avatar.
NPI is conditional show for pharmacists only; validate on submit.

Invites and security model

Invite tokens are high-entropy and opaque. Only the SHA-256 hash is stored in the database.

Schema
invites: inviteId, email, companyId, role, tokenHash CHAR(64) UNIQUE, tokenUsed TINYINT DEFAULT 0, expirationTime DATETIME DEFAULT (CURRENT_TIMESTAMP + INTERVAL 72 HOUR), inquiryId nullable, createdAt.
users: id, email, role (prefer slugs: admin | manager | employee), companyId, firstName, lastName, avatarUrl nullable.
companies: companyId, name, npi, address...

Token utilities
makeToken() → base64url string.

hashToken(token) → sha256 hex string.
Email the plain token inside the link; never store it.

DAO pattern
invitesDao.getOnboardingContextByTokenHash(tokenHash) performs a single SELECT with LEFT JOINs to companies and inquiries, with WHERE tokenUsed = 0 AND expirationTime > NOW().
Return one row or null.

GET /api/onboarding
Reads token from req.query, hashes it, looks up the invite context, and returns JSON { role, company { id, name, npi }, prefill { email, firstName, lastName } }.
Return 400 on missing/invalid token shape; 410 on expired or not found.

POST /api/onboarding/complete
Hash and re-lookup the invite with the same WHERE conditions.
Validate required fields server-side.
Create user using invite.role and invite.companyId; ignore client-sent role or companyId.
Start session.
Mark invite used atomically within a transaction.

First-manager rule
When approving an inquiry, create the first invite with role manager.
Managers may invite manager or employee for their own company.
Admins may invite any role.
Guard: if a company has no manager user and no valid manager invite, reject non-manager invite creation.

Avatar upload plan

Endpoint: POST /api/users/me/avatar (requireAuth).
Multer memoryStorage to accept the file.
Sharp to rotate by EXIF, square-crop and resize to 256 and 64, encode as WebP, strip metadata.
Serve /uploads and store users.avatarUrl.
DELETE /api/users/me/avatar removes and clears avatarUrl.

Shared form utilities

form-ui.js exposes
setSubmitBusy(btn, busy) for disabled state, classes, and aria-busy.
checkSubmit(form, btn, predicate) to enable the button only when valid.
onSubmit(form, handler) to pass FormData without page reload.
wireForm combines the above for convenience.
Buttons also gray on first paint via CSS rule .btn-primary[disabled].

Email links and UX polish

Use base64url tokens so links are URL-safe.
Add meta noindex and referrer no-referrer on onboarding.html.
Never log query strings containing tokens.
If token missing, redirect with location.replace to /index.html or show a small error with a go-home action.

Testing checklist

Thunder Client
GET /api/onboarding with token param
Expect 200 for valid, 410 with { error: 'expired' } for expired, 410 with { error: 'invalid' } for wrong token.
POST /api/onboarding/complete with FormData including token
Expect 201, a session cookie, and invites.tokenUsed = 1.

Coding conventions

Keep readable slugs for roles in API responses admin | manager | employee.
Prefer explicit LEFT JOINs over NATURAL JOINs for stability.
Validate critical conditions in SQL where possible expirationTime > NOW(), tokenUsed = 0.
Keep public pages and scripts under public; secure pages under secure.
Use type="module" for all page scripts.

Open tasks

Finalize onboarding complete endpoint and DB transaction.
Add Change email flow during onboarding.
Build Invites tab in Admin: list, resend, revoke.
Implement avatar endpoints with Multer + Sharp.
Add body role classes in main.js and hide admin link by default.
Switch users.role to slugs or a roles table.
Add tiny metrics to dashboard and course placeholders.