🔍 Full Project Review: YEHA Learnership Management System

---

1. 📁 Project Structure

Stack Summary

This is a **full-stack Next.js 14** monorepo using the App Router pattern, with **Prisma + SQLite** as the database, **Tailwind CSS** for styling, and **TypeScript** throughout.

Structure Rating: ⚠️ 5/10 — Needs Cleanup

✅ What's Good:

- `src/app/` uses the Next.js App Router convention correctly — pages and API routes are co-located
- `src/components/`, `src/hooks/`, `src/contexts/`, `src/lib/`, `src/types/` are properly separated
- Prisma schema lives in `prisma/schema.prisma` — correct placement
- Frontend and backend are co-located in the monorepo as expected for Next.js

🚩 Critical Problems:

Issue	Details
~28 orphan log/dump files in root	`build_error.log`, `build_output_*.txt`, `lag_dump.txt`, `server.log`, `upload-*.txt`, etc. are cluttering the root directory
~30+ orphan scripts in root	Files like `check-db.js`, `add-montazility.js`, `test-login.js`, `debug-auth.js`, `make-admin.js` should be in `scripts/`
~20+ orphan Markdown docs in root	`ATTENDANCE_ASSESSMENT_FIXES.md`, `BUILD_FIX_COMPLETE.md`, `PHASE2_MASTER_PROMPT.md` etc. — should live in a `docs/` folder
Duplicate/leftover files	`CourseCreationForm_NEW.tsx`, `ProgressReport_NEW.tsx`, `StudentContext_OLD.txt` — dead code left behind from refactors
`.html` test files in root	`test-api.html`, `test-api-direct.html`, `test-attendance.html` — should be in a `tests/` folder or removed
14MB `dev.db` in tracked folder	`prisma/dev.db` is **NOT .gitignored** — this SQLite DB should never be committed
`lib/` at root AND in `src/`	There's a `lib/` directory at the project root with 1 child — likely orphaned

Issue	Details
`Skills Folder/` and `Roll Out/`	Non-standard directories with spaces in names, sitting at root

2. 🎨 Frontend Review

Stack: **Next.js 14 + React 18 + TypeScript + Tailwind CSS + Lucide React + Recharts + SWR**

Component Organisation: ⚠️ 6/10

✅ **What's Good:**

- Components are reasonably modular — 55 component files + 7 UI primitives + 1 AI component
- Uses `dynamic()` imports for heavy dashboard components (`DashboardCharts`, `RecentActivity`)
- Good use of `Suspense` boundaries with loading skeletons on `src/app/page.tsx`
- Custom hooks (`useDashboard.ts`, `useStudents.ts`, etc.) properly abstract API calls using SWR
- AuthContext, GroupsContext, and StudentContext properly wrapped via `Providers` component
- Google Fonts (Outfit + Lora) properly loaded with CSS variables
- ErrorBoundary component wraps the entire app

🚩 **Problems:**

File	Issue
`src/components/CourseCreationForm_NEW.tsx`	Duplicate of `CourseCreationForm.tsx` — dead code
`src/components/ProgressReport_NEW.tsx`	Identical size (9,093 bytes) to `ProgressReport.tsx` — dead code

File	Issue
`src/contexts/StudentContext_OLD.txt`	Leftover backup file with `.txt` extension
`src/contexts/StudentContextSimple.tsx`	Confusing name — both `StudentContext.tsx` and `StudentContextSimple.tsx` exist but only `Simple` is used in `layout.tsx`
55 flat component files	No sub-folders (e.g., `components/modals/`, `components/dashboard/`, `components/forms/`) — all 55 files dumped in one directory
Giant component files	`GroupsManagement.tsx` (31KB), `StudentDetailsModal.tsx` (26KB), `EditStudentModal.tsx` (18KB) — these should be broken into smaller sub-components
`src/app/page.tsx` line 61	`icon: any` — weak typing, should be `icon: LucideIcon`
No `<head>` meta per page	Only the root layout has metadata; individual pages don't export their own `metadata` objects

Accessibility: ⚠️ 4/10

- No `aria-label` attributes found on interactive elements
- No keyboard navigation enhancements visible
- No skip-to-content link
- Table headers use appropriate `<th>` elements ✅

---

3. 🔧 Backend Review

Stack: **Next.js API Routes + Prisma ORM + SQLite + bcryptjs + jose (JWT) + Zod validation**

API Structure: ✅ 7/10

✅ What's Good:

- 29 API route groups covering all domains (auth, students, groups, assessments, attendance, etc.)
- `src/lib/prisma.ts` — correct singleton pattern for PrismaClient in development
- `src/lib/auth.ts` — JWT authentication using `jose` (edge-compatible)
- `src/lib/validation.ts` — comprehensive Zod schemas for all major entities
- `src/lib/input-sanitizer.ts` — XSS prevention utilities
- `src/lib/rate-limit.ts` — in-memory rate limiting (functional but basic)
- API routes use `try/catch` wrapping extensively ✅

🚩 Security Issues:

Severity	File	Line	Issue
🔴 **CRITICAL**	`.env`	8	**JWT secret hardcoded**: `JWT_SECRET="yeha-learnership-secret-key-2026"` — this is a real, guessable secret checked into version control
🔴 **CRITICAL**	`.env`	18-20	**API keys exposed**: Cohere, Pinecone, and ZAI API keys in `.env` which IS gitignored, but...
🔴 **CRITICAL**	`.env.local`	14, 17	**Different API keys in `.env.local`** — ALSO contains real Pinecone/Cohere keys. `.env.local` IS gitignored ✅ but `.env` is also gitignored. However, the file still **exists physically** and could have been committed previously
🔴 **CRITICAL**	`.env`	11-12	**Supabase URL + anon key** exposed with full JWT

Severity	File	Line	Issue
🟠 **HIGH**	`src/lib/auth.ts`	12	**Hardcoded fallback secret**: `'yeha-learnership-secret-key-2026'` — if `JWT_SECRET` env var is missing, this weak secret is used
🟠 **HIGH**	`src/middleware.ts`	47-54	**ALL attendance endpoints are unprotected** — any unauthenticated user can POST/PUT/DELETE to `/api/attendance/*`
🟠 **HIGH**	`src/middleware.ts`	35-45	**GET requests to `/api/groups` and `/api/students` bypass auth** — leaks all student/group data
🟡 **MEDIUM**	`src/middleware.ts`	26	**`console.log` in middleware** — logs every request in production
🟡 **MEDIUM**	`src/lib/rate-limit.ts`	8	In-memory rate limiting doesn't survive restarts and has no distributed support
🟡 **MEDIUM**	`prisma/dev.db`	—	**14MB SQLite database file** not in `.gitignore` — could be committed to Git with real student data

Database: ⚠️ 6/10

Issue	Details
SQLite in production	The project uses SQLite (`file:./dev.db`), which is fine for development but not suitable for production deployment
Dev.db not gitignored	The `.gitignore` only ignores `.env*.local` and `.env` — the actual database file `prisma/dev.db` is not excluded
No migration files	There are no `prisma/migrations/` directory — suggests the schema is pushed directly with `prisma db push` rather than using versioned migrations
Good schema design ✅	Well-structured with proper relations, indexes, and unique constraints

4. 🔌 API & Data Flow

Rating: ✅ 7/10

✅ **What's Good:**

- Frontend hooks (`useStudents`, `useDashboard`, etc.) use SWR for data fetching with proper error/loading states
- API response format is consistent: `{ data: {...} }` or `{ success: false, error: '...' }`
- Proper `mutate()` for cache invalidation after mutations
- Global SWR config (`src/lib/swr-config.ts`) with deduplication and refresh intervals

🚩 **Problems:**

File	Issue
`src/app/page.tsx` lines 121-133	Dashboard fetches from `/api/dashboard/summary` using raw `fetch()` instead of the existing `useDashboardStats` SWR hook — inconsistent pattern, results in duplicate fetching
`src/hooks/useDashboard.ts` vs `src/hooks/useDashboardStats.ts`	Two separate hooks for dashboard data — `useDashboardStats.ts` and `useDashboard.ts` both export `useDashboardStats()`
`src/app/page.tsx` line 124	No error handling for non-OK responses — `response.ok` check exists but no user-facing error state is set
`src/contexts/AuthContext.tsx` line 37	`JSON.parse(storedUser)` — no try/catch around localStorage parse; corrupted data will crash the app

---

5. 🐛 Bugs & Errors

#	Severity	File	Line	Issue
1	🔴 HIGH	`src/contexts/AuthContext.tsx`	37	`JSON.parse(storedUser)` without try/catch — corrupted localStorage will crash the entire app on load
2	🔴 HIGH	`src/middleware.ts`	47-54	All attendance routes (POST/PUT/DELETE) are completely unprotected — anyone can modify attendance data
3	🟡 MEDIUM	`src/lib/auth.ts`	12	Fallback JWT secret hardcoded — in any environment where `JWT_SECRET` is unset, tokens are signed with a predictable key
4	🟡 MEDIUM	`src/app/page.tsx`	61	`icon: any` — weak TypeScript typing
5	🟡 MEDIUM	`src/hooks/useDashboardStats.ts` + `useDashboard.ts`	—	Two hooks both export `useDashboardStats`, creating import ambiguity
6	🟡 MEDIUM	`next.config.mjs`	11	`eslint.ignoreDuringBuilds: true` — ESLint is completely disabled during builds, masking potential bugs
7	🟢 LOW	`src/lib/rate-limit.ts`	51-58	`setInterval` at module scope — runs even during build/SSR; no cleanup mechanism
8	🟢 LOW	`src/app/page.tsx`	119	React hook dependency: `fetchDashboardData` is not in the dependency array of `useEffect`, may cause stale closures

---

6. 🔐 Environment & Config

`.env` Files:

File	Gitignored?	Contains Secrets?	Status
`.env`	✅ Yes (`.env` in `.gitignore`)	✅ **YES — JWT, Supabase, Cohere, Pinecone, ZAI keys**	⚠️ May have been committed before `.gitignore` was updated
`.env.local`	✅ Yes (`.env*.local`)	✅ **YES — Different Pinecone/Cohere keys**	⚠️ Has different keys than `.env` — confusing
`.env.example`	❌ Not gitignored (but only has placeholders)	❌ No real values	✅ Good — but outdated; doesn't list Pinecone/Cohere/ZAI vars

Critical Environment Issues:

Issue	Details
Conflicting `.env` vs `.env.local`	`DATABASE_URL` in `.env` is an absolute Windows path; in `.env.local` it's `file:./dev.db`. `JWT_SECRET` differs between them. Different API keys for Pinecone/Cohere. This is confusing and error-prone.
`.env.example` is outdated	Doesn't list `PINECONE_API_KEY`, `COHERE_API_KEY`, `ZAI_API_KEY`, `RESEND_API_KEY`, `CRON_SECRET`, or `NEXT_PUBLIC_APP_URL`
`dev.db` not gitignored	SQLite database with real student data could be committed
Absolute path in `DATABASE_URL`	`.env` line 4: `file:C:\\Users\\LATITUDE 5400\\...` — won't work on any other machine

---

7. 📖 README & Documentation

Rating: ⚠️ 4/10

✅ **What's Good:**

- Has a README with installation steps, project structure, and troubleshooting
- Lists tech stack and available scripts

🚩 **Problems:**

Issue	Details
Massively outdated	README describes this as a "Static Frontend" (Phase 1), says backend is "Planned" — but the project already has full backend, auth, Prisma, AI integration, 29 API route groups
Missing database setup	No mention of `npx prisma generate`, `npx prisma db push`, or `npx prisma db seed` — database won't work from a fresh clone
Missing `.env` setup	Doesn't explain which environment variables are required
Project structure diagram is wrong	Shows only `Sidebar.tsx` and `Header.tsx` under components — actual project has 55+ components
No mention of AI features	Cohere, Pinecone, ZAI integrations are not documented
Claims "Static Export"	Line 101-109 describes deployment as static HTML — but the app requires a Node.js server for API routes
Too many competing docs	20+ markdown files in root (`START_HERE.md`, `COMPLETE_SITEMAP.md`, `FEATURE_GUIDE.md`, etc.) — no clear entry point

---

8. 🏆 Top Improvements (Priority Order)

🔴 Must Fix (Security & Correctness)

#	Action	Why
1	**Rotate ALL exposed API keys** (Cohere, Pinecone, ZAI, Supabase)	If this repo was ever pushed to GitHub, all keys are compromised
2	**Remove hardcoded JWT fallback** in `src/lib/auth.ts:12`	Makes the auth system predictable if env is misconfigured

#	Action	Why
3	**Protect attendance endpoints** in middleware	Currently anyone can manipulate attendance records
4	**Add `prisma/dev.db` to `.gitignore`**	Prevent leaking student PII data
5	**Add try/catch to `JSON.parse`** in `AuthContext.tsx:37`	Prevents app-crashing errors
6	**Remove `console.log` from middleware**	Noise in production, minor performance hit per request

🟡 Should Fix (Code Quality)

#	Action	Why
7	**Delete all orphaned files from root** — logs, scripts, test HTML	Reduces cognitive load, clutter
8	**Delete `_NEW` and `_OLD` files**	Dead code creates confusion
9	**Update `.env.example`** with all required vars	New developers can't set up the project
10	**Rewrite README.md** to reflect actual project state	Documentation is misleading
11	**Consolidate `useDashboard.ts` and `useDashboardStats.ts`**	Duplicate hooks cause confusion
12	**Organize components into sub-folders**	55 flat files are hard to navigate
13	**Re-enable ESLint in builds** (`next.config.mjs`)	Hiding lint errors lets bugs accumulate

🟢 Nice to Have (Professional Polish)

#	Action	Why
14	**Migrate SQLite → PostgreSQL** for production	SQLite doesn't support concurrent writes and can't be deployed to serverless

#	Action	Why
15	**Add Prisma migrations** (not just `db push`)	Versioned schema changes are essential for production
16	**Add accessibility improvements** (aria labels, skip nav, focus rings)	Accessibility compliance
17	**Split giant components** (GroupsManagement, StudentDetailsModal)	Maintainability
18	**Add distributed rate limiting** (Redis-based)	In-memory map won't work across instances
19	**Add API response typing**	Several hooks use loose typing
20	**Consolidate root markdown docs** into `docs/` directory	Clean project root

Summary Scorecard

Category	Score	Notes
Project Structure	5/10	Correct patterns but massive clutter
Frontend	6/10	Good React patterns, needs cleanup
Backend	6/10	Solid API design, serious security gaps
API & Data Flow	7/10	SWR hooks are well-done, some inconsistencies
Security	3/10	Exposed keys, unprotected routes, hardcoded secrets
Environment & Config	4/10	Conflicting env files, outdated example
Documentation	4/10	Severely outdated README
Overall	**5/10**	Functional but needs significant hardening before production