Kicking off a comprehensive project report so you have a clear overview of what’s built, how it’s wired, and where to go next.

Plan -

- Summarize the product goals and scope.

- Detail the architecture and technology stack (frontend, backend, database, real-time).

- Map core features to implementation.

- Explain security/auth and multi-tenancy.

- Provide a file-by-file guide (focused and actionable).

- Include build/run steps and operational notes.

- List known gaps, warnings, and next steps.

Executive Summary

SmartAttend is a Next.js-based attendance platform with:

- Role-based access: student, teacher, admin, superadmin.

- Multi-tenancy via institutes to isolate data/admin control.

- Hybrid data layer: MongoDB (via `mongoose`) for structured models; CouchDB (via `nano`) for document-style analytics and durable logs/exports.

- Biometric and geolocation factors: face descriptors (face-api.js), proximity/geofence checks.

- Real-time updates via Socket.IO with optional SSE fallback.

- Admin dashboards: approvals, courses/schedules, analytics, manual attendance.

- Teacher tools: create session, QR join workflow, live attendance.

- Student UX: onboarding, profile, join session (camera + QR + biometric).

- Theming and UI: dark/light themes with CSS variables and modern glass UI.

Architecture Overview

- Frontend: Next.js 15 (pages router), React 19, Tailwind v4 via `@tailwindcss/postcss`, Chart.js + react-chartjs-2, Recharts, lucide-react icons, framer-motion for micro-interactions.

- Backend:

  - Next.js API routes (server-side session checks with NextAuth JWT).

  - MongoDB (Mongoose) for primary entities.

  - CouchDB (nano) for analytics/backup/export-friendly docs.

  - Socket.IO for real-time messaging (server route + client hooks).

- Security:

  - Credentials auth via NextAuth; hashed passwords with `bcryptjs`.

  - Biometric face descriptors via face-api.js (vector distance check).

  - Proximity/geofence validation.

  - Token/session validation patterns (expansion ready).

- PWA:

  - manifest.json, sw.js, basic headers in next.config.js.

Technology Stack

- Core: `next@15`, `react@19`, `react-dom@19`

- Auth: `next-auth@4`, `bcryptjs`

- DB: `mongoose@8` (MongoDB), `nano@11` (CouchDB)

- Real-time: `socket.io`, `socket.io-client`

- Biometrics: `face-api.js`

- QR: `jsqr` (scanner), `qrcode.react` (render)

- Charts: `chart.js`, `react-chartjs-2`, `recharts`

- UI: Tailwind v4, `framer-motion`, `lucide-react`

Key Features and Implementation Mapping

- Roles and Access Control
  - NextAuth Credentials provider with JWT strategy.
  - Server-side session retrieval on APIs and SSR pages.
- Multi-tenancy
  - Institute-based scoping across admin/teacher/student flows.
- Student Onboarding and Approvals
  - Admin approvals pages and APIs.
  - Face descriptor capture and approval.
- Attendance
  - Session creation with validation requirements (proximity/geofence/biometric).
  - QR-based join: student camera + QR decode + face capture.
  - Manual attendance fallback and finalize-session (absent marking).
- Analytics and Dashboards
  - Charts for departments, subjects, course performance.
  - Real-time teacher dashboard updates via Socket.IO.
- Theming and UI
  - Global CSS variables, glassmorphism, explicit dark/light theme toggle.

Authentication & Authorization

- File: auth.js
  - NextAuth with Credentials provider.
  - Looks up user by email within selected role (`Student`, `Teacher`, `Admin`).
  - Uses `bcryptjs` for password verification.
  - JWT stores `id` and `role`; session callback exposes to `session.user`.

- Pages using session:

  - Admin/teacher/student dashboards and their SSR data loaders require the session server-side (`getServerSession`) where appropriate.

Required secrets/env:

- `NEXTAUTH_SECRET` (for NextAuth)

- `MONGODB_IP` (MongoDB connection string; see note below)

- `COUCHDB_URL` (CouchDB URL, set/exposed via next.config.js serverRuntimeConfig)

Note: database.js reads `MONGODB_IP` from `serverRuntimeConfig` via `next/config`. Ensure next.config.js exposes it similarly to `COUCHDB_URL`, or refactor database.js to read `process.env.MONGODB_IP` directly.

Data Layer

- MongoDB via Mongoose

  - Schemas in `src/models/*.js`:

    - Student.js: Email/password (hashed), role, academic info, courses.

    - Teacher.js: Identity, institute/department fields (as defined).

    - Admin.js: Admin identity and role.

    - Course.js: Course meta, referenced by students/sessions.

    - Session.js: Course/time references; used for attendance sessions.

    - Attendance.js: Student/session status (present/left_early/absent), timestamps.

  - Database connection: database.js (cached connection across hot reloads).

- CouchDB via Nano

  - Connector: couchdb.js (ensures DB exists, caches connection).

  - Used in many API routes to store and query analytics/approval and other documents under a single DB (`smartattend`).

Real-time

- Server: websocket.js (Socket.IO server creation).

- Client utilities: realTimeUpdates.js (connect/broadcast helpers).

- Used to broadcast:

  - Attendance updates.

  - Session state changes.

  - Security alerts (biometric mismatch, etc.).

- SSE fallback under [sessionId].js (route exists per build output).

Biometrics & QR Join

- Student: join-session.js

  - Uses `getUserMedia` to open camera.

  - QR scanning via `jsqr` on image frames (offscreen canvas).

  - Dynamically imports `face-api.js`, loads models from models.

  - Captures a face descriptor and submits attendance with net info + descriptor.

- Admin Approvals: approvals.js

  - Dynamically imports `face-api.js` on demand to process descriptors for review.

Theming & UI

- Global CSS: globals.css

  - CSS variables for theme tokens, glass backgrounds, borders.

  - Explicit dark/light handling via `html[data-theme=...]`.

- Toggle: ThemeToggleButton.js

  - Switches `data-theme`, persists preference, minimal visuals.

- Charts:

  - `src/components/analytics/*.js` (bar, pie, line) using Chart.js and Recharts with theme-aware options.

PWA and Assets

- manifest.json — app manifest.

- sw.js — service worker.

- next.config.js sets headers for `sw.js` and `manifest.json` and a rewrite for service worker path.

API Overview (routes per build)

Dynamic server-rendered or API routes relevant to core features:

- Auth:

  - [...nextauth].js

  - register.js

- Admin:

  - approvals.js

  - index.js, [id].js

  - course-students.js

  - session-attendance.js

  - manual-attendance.js

  - finalize-session.js

  - register-admin.js, register-teacher.js, bootstrap endpoints

  - analytics.js

  - analytics.js

  - `src/pages/api/admin/users.js`, [id].js, `users/reset-password.js`
  - analytics.js, `export-analytics.js`, `predictive-analytics.js`
- Teacher:
  - session.js (create session; set validation policy)
  - session-attendance.js
  - manual-attendance.js
  - finalize-session.js
  - analytics.js, students.js
- Student:
  - mark-attendance.js (validates proximity/geofence/biometric)
  - validate-session.js
  - profile.js, `student/onboarding.js`, analytics.js
- Realtime:
  - websocket.js
  - [sessionId].js
- Misc:
  - contact.js (present in build manifest)
  - debug-session.js

Pages and Components (selected file roles)

- Pages root:
  - index.js: Landing page (marketing content, CTA).
  - login.js: Credentials login with role selection, responsive.
  - register.js: Registration entry point (admin/student paths).
  - dashboard.js: Post-login role router or shared dashboard view.

  - logout.js: Simple sign-out.

- Admin

  - dashboard.js: Admin overview, shortcuts (courses, approvals, analytics).

  - courses.js: CRUD UI for courses/schedules.

  - approvals.js: Student face approvals, search/recent/retake.

  - institutes.js: Multi-tenancy institute management (superadmin access).

  - register-admin.js / register-teacher.js: Superadmin admin-creation; admin teacher-creation.

  - [id].js, [id].js, [id].js: Detail dashboards and analytics.

- Teacher

  - create-session.js: Create session, generate QR via `qrcode.react`.

  - dashboard.js: Live attendance and session management.

- Student

  - onboarding.js: Capture profile info and unique identifiers, face scan setup.

  - join-session.js: QR scan + face capture + submit.

  - profile.js, dashboard.js.

- Components

  - FaceScanner.js: Camera capture and face-api.js helpers (if present).

  - RealTimeTeacherDashboard.js: Live session stats, uses `react-intersection-observer` for animations/triggers.

  - ThemeToggleButton.js: Theme control with framer-motion, lucide icons.

  - Analytics (Chart.js):

  - CoursePerformanceBarChart.js, HistoryGraph.js

  - DepartmentPieChart.js, SubjectPieChart.js

  - StatsCard.js, StatusPieChart.js (mixes Recharts where appropriate)

- Lib

  - auth.js: NextAuth configuration.

  - database.js: Mongo connection (MONGODB_IP).

  - couchdb.js: CouchDB nano client (COUCHDB_URL).

  - api-helper.js, `auth-helper.js`: Common API helpers (if present in your tree).

  - auth.js: Credential checks against Mongoose models.

- Utils

  - realTimeUpdates.js: Socket.IO utilities and fallbacks.

  - proximityValidation.js: Proximity/geofence logic (ensure this aligns with mark-attendance).

  - sessionValidation.js: Token generation/validation (if present).

- Public

  - `public/models/...`: face-api.js model manifests and shards.

  - sw.js, manifest.json, offline.html: PWA assets.

Security and Privacy

- Passwords: `bcryptjs` for hashing; `StudentSchema.pre('save')` re-hashes on change, `matchPassword` for comparison.

- Face data: stores descriptors (embedding vectors), not raw images.

- Sessions: NextAuth JWT strategy; role and ID encoded in token claims.

- Route protection: Server-side session checks in API and SSR loaders.

- Geofence and proximity: Validates IP/WiFi and optional location radius, plus biometric matching (typical distance threshold around 0.6 for face-api.js embeddings).

DevOps and Scripts

- Scripts found under scripts:

  - backup-db.js: Export CouchDB documents to timestamped JSON.

  - restore-db.js: Bulk restore JSON into CouchDB (preserving rev history with `new_edits=false`).

  - migrate-db.js: Reshape user docs to nested structure, add `instituteId`, bump `schemaVersion`.

- Package scripts:

  - `dev`, `build`, `start`, `lint`

  - `db:backup`, `db:migrate`, `db:restore`, plus placeholders for audits/exports.

Dependency Hygiene

Recent updates:

- Added: `jsqr` (required by join-session.js).

- Removed unused: `@gsap/react`, `gsap`, `lottie-react`, `@yudiel/react-qr-scanner`, `micro`, `chartjs-adapter-date-fns`, `date-fns`, `next-themes`, `qrcode`.

- Build validated after changes.

Security audit:

- `npm install` reports 3 vulnerabilities (2 low, 1 high). Consider `npm audit` and targeted updates (avoid `--force` unless acceptable).

Build & Run (Windows cmd.exe)

Set required environment variables (sample):

```
set NEXTAUTH_SECRET=your-strong-secret
set MONGODB_IP=mongodb://localhost:27017/smartattend
set COUCHDB_URL=http://admin:password@localhost:5984
```

Install, build, run:

```
npm install
npm run build
npm start
```

Development:

```
npm run dev
```

CouchDB backup/restore/migrate (examples):

```
set COUCHDB_URL=http://admin:password@localhost:5984
node scripts/backup-db.js
node scripts/migrate-db.js
node scripts/restore-db.js path\to\backup.json
```

```

## Known Warnings and Operational Notes

- ESLint warning: .eslintignore is deprecated in the new flat config. Migrate ignores to eslint.config.mjs.

- Ensure `MONGODB_IP` is available to database.js. Currently it reads from `serverRuntimeConfig`; either:

  - Add `MONGODB_IP: process.env.MONGODB_IP` to next.config.js `serverRuntimeConfig`, or

  - Update database.js to use `process.env.MONGODB_IP` directly on the server.

- Large face-api.js models must be hosted under models as configured; ensure paths match your dynamic loader in `join-session` and `approvals`.

## Roadmap and Next Steps

- Resolve `eslint` ignore deprecation.

- Address `npm audit` findings with safe upgrades.

- Optional: consolidate storage to a single DB (Mongo or Couch) if desired; otherwise clearly document what lives where.

- Add in-app surface for real-time security alerts (admin/teacher panes).

- Expand tests (unit tests for validators, token logic, critical API routes).

- Consider privacy review/documentation for biometric data handling.

## Appendix: File Map (selected)

- Config

  - next.config.js: Runtime config (COUCHDB_URL), headers, rewrites, dev tweaks.

  - eslint.config.mjs, postcss.config.mjs, jsconfig.json.

- Public
```

  - manifest.json, sw.js, offline.html, `public/models/*`.

- Lib

  - auth.js: NextAuth provider/callbacks.

  - database.js: Mongoose connector.

  - couchdb.js: Nano connector.

- Models (Mongoose)

  - Admin.js, Attendance.js, Course.js, Session.js, Student.js, Teacher.js, `schemas.js`.

- Utils

  - realTimeUpdates.js, `networkDetection.js`, `proximityValidation.js`, `sessionValidation.js`, `gamification.js`.

- Components

  - ThemeToggleButton.js, `FaceScanner.js`, RealTimeTeacherDashboard.js.

  - `src/components/analytics/*`: charts and stat cards.

- Pages

  - Root: `_app.js`, index.js, login.js, `logout.js`, dashboard.js, register.js

  - Auth: `pages/api/auth/[...nextauth].js`, register.js

  - Admin: `admin/*.js`, `api/admin/*`

  - Teacher: `teacher/*.js`, `api/teacher/*`

  - Student: `student/*.js`, `api/student/*`

  - Realtime/Streaming: websocket.js, `api/sse/[sessionId].js`

Completion Summary

- Delivered a full project report covering architecture, stack, features, security, API/page map, and ops.

- Cleaned package.json (removed unused, added `jsqr`) and verified builds succeed.

- Provided clear Windows-friendly run commands and flagged config/env requirements.

If you want, I can:

- Patch next.config.js to expose `MONGODB_IP` (or refactor database.js).

- Migrate ESLint ignores to the new flat config.

- Run `npm audit` and propose non-breaking remediations.