# README.md

# EHR Integration Dashboard — Final Deliverable

Project: EHR CRUD Dashboard (DrChrono integration — starter implementation)
Prepared for: Assignment submission
Prepared by: Chinni Krishna (use/edit as needed)

This archive contains:
- Implementation_Guide.md — detailed architecture, auth flows, state and error handling, HIPAA notes
- API_Discovery.md — endpoints used, sample requests/responses, capabilities & limitations
- postman_collection.json — example Postman collection (auth, patients, appointments)
- nextjs_starter/ — minimal Next.js + TypeScript starter scaffold (API proxy, types, hooks)
- tests/ — example unit test for token refresh logic
- .env.example — environment variables required
- email_template.txt — ready-to-send email with a link to this ZIP

How to use:
1. Download and unzip this archive.
2. Fill `.env` variables from `.env.example`.
3. Run `npm install` inside `nextjs_starter` and `npm run dev` to start the dev server.
4. Use the Postman collection to exercise the API calls (or import into Postman).

Notes:
- No production credentials are included. Replace placeholders in `.env` with your DrChrono/Epic test app credentials.
- This deliverable is a starter scaffold plus full documentation for your assignment submission.

# Implementation_Guide.md

# Implementation Guide — EHR Integration Dashboard

## Overview
This document explains how the integration works, architecture decisions, authentication, state management, error handling, and deployment recommendations for the EHR CRUD Dashboard assignment.

Chosen EHR: DrChrono (REST API). The deliverable can be adapted to Epic (SMART-on-FHIR) by replacing the client flow with SMART launch and FHIR resources.

## Architecture
- **Frontend:** Next.js (TypeScript). Pages for Patients, Appointments, Clinical, Billing, Settings.
- **Server / Integration layer:** Next.js API routes used as a secure proxy to call DrChrono APIs. Keeps client secrets server-side.
- **Optional DB:** PostgreSQL for token storage (encrypted), audit logs, app users & RBAC.
- **Auth:** OAuth2 Authorization Code (server-side). Access tokens + refresh tokens stored encrypted server-side.

Sequence (simplified):
1. User clicks "Connect EHR" -> redirects to DrChrono OAuth authorize URL.
2. DrChrono redirects back to `/api/auth/callback?code=...`.
3. Server exchanges `code` for access + refresh token; stores tokens encrypted.
4. Frontend calls Next.js API routes; server attaches `Authorization: Bearer <access_token>` when calling DrChrono.

## Key Implementation Details

### OAuth & Token Management
- Use Authorization Code flow with server-side exchange.
- Store tokens encrypted at rest (e.g., KMS or DB field encryption).
- Implement token refresh endpoint and proactively refresh before expiry.
- Do not expose client_secret on the frontend.

### Proxy Pattern (Security)

- All calls to EHR should go through server-side proxy API routes.
- Proxy enforces RBAC, records audit logs, and sanitizes responses to remove any accidental logs of PHI.

### State Management
- **React Query (TanStack)** for server state (patients, appointments).
- **React Context + useReducer** for session, user roles, and app-level settings.
- Keep PHI out of localStorage; use HttpOnly cookies for session tokens.

### Error Handling Strategy
- Standardize error response format: `{ code, message, details? }`.
- For upstream 401: attempt token refresh once; if refresh fails, require re-auth.
- For rate-limits (429): exponential backoff with jitter; surface friendly UI message.
- Log server-side errors with correlation IDs for tracing.

### Performance Optimizations
- Use pagination and server-side filtering for patient lists.
- Cache common responses with React Query; use `stale-while-revalidate` patterns.
- Use bulk export endpoints for large reports.

### Audit & Compliance
- Record user id, action type (READ/WRITE/DELETE), target resource id, timestamp, and client IP.
- Keep audit logs immutable if possible and retain according to org policy.
- Enforce TLS, encrypt tokens & backups, and implement RBAC with least privilege.

## Development & Deployment
- Use Vercel for Next.js hosting (automatic HTTPS).
- Store secrets in Vercel Environment Variables or a secrets manager.
- CI pipeline: run tests, lint, build; deploy to preview environment.

## Files of interest in this package
- `nextjs_starter/pages/api/auth.ts` — OAuth skeleton
- `nextjs_starter/pages/api/proxy/[...path].ts` — Proxy skeleton
- `API_Discovery.md` — endpoint list used for the assignment
- `postman_collection.json` — importable into Postman

## Known limitations & notes
- Some lab write endpoints or billing features may require DrChrono enablement; contact DrChrono support for vendor-specific access.
- Epic integration requires implementing SMART-on-FHIR and adapting resource models to FHIR R4.

# API_Discovery.md

# API Discovery — DrChrono (selected endpoints)

This document lists the endpoints used for the CRUD dashboard demo along with sample requests/responses and notes.

## Patient Management
- `GET https://drchrono.com/api/patients`
  - Query params: `first_name`, `last_name`, `page`, `limit`, `search`
  - Sample response (trimmed):
  ```json
  {
    "meta": { "next": null },
    "results": [
      {
        "id": 12345,
        "first_name": "John",
        "last_name": "Doe",
        "date_of_birth": "1980-02-14",
        "phone_number": "555-1234",
        "email": "john@example.com",
        "medical_record_number": "MR-001"
  ```

```
    }
  ]
}
```

- `GET https://drchrono.com/api/patients/{id}` — retrieve patient details
- `PUT https://drchrono.com/api/patients/{id}` — update demographics/contact (payload: only updatable fields)
- `POST https://drchrono.com/api/patients` — create new patient (depends on account permissions)

## Appointments
- `GET https://drchrono.com/api/appointments`
  - Params: `date`, `doctor`, `patient`, `page`
- `POST https://drchrono.com/api/appointments` — create (requires provider id, patient id, start time)
- `PUT https://drchrono.com/api/appointments/{id}` — reschedule or update
- `DELETE https://drchrono.com/api/appointments/{id}` — cancel

## Clinical
- `GET https://drchrono.com/api/notes?patient={patient_id}` — fetch notes
- `POST https://drchrono.com/api/notes` — add clinical note
- `GET https://drchrono.com/api/lab_results?patient={id}` — lab results (read)

## Billing & Admin
- `GET https://drchrono.com/api/transactions?patient={id}` — payments / balances
- `GET https://drchrono.com/api/eligibility?patient={id}` — insurance eligibility (if available)

## Common patterns & headers
- Authorization: `Bearer <access_token>`
- Content-Type: `application/json`

## Limitations & Notes
- Pagination: many endpoints use paginated results with `next` links.
- Rate limits: handle 429 with backoff.
- Some write operations (labs, billing) may require additional account privileges.

# email_template.txt

Subject: EHR Integration Dashboard — Submission

Hi [Recipient Name],

Please find attached the final deliverable for the EHR Integration Dashboard assignment.
I have included:
- A Next.js TypeScript starter scaffold (server-side proxy + client hooks)
- Implementation guide and API discovery document
- Postman collection and example unit tests

Download the ZIP here: [ATTACH ZIP OR INSERT LINK]

Notes:
- Replace placeholders in `.env.example` with your DrChrono/Epic credentials.
- The project uses a proxy pattern—serverless API routes attach access tokens securely.

Thanks,
Chinni Krishna