MERN Rule-Based System Design for Health Report Analysis

Frontend (React / Next.js + Tailwind)

- Upload UI (file + metadata), Dashboard for biomarker cards, charts, alerts.
- Admin Rule Editor for clinicians to create/test rules.
- APIs: POST /api/reports/upload, GET /api/reports/:id, GET /api/rules.
- Charts with Recharts/Chart.js, Explainability modal for fired rules.

Backend (Node.js + Express, Rule Engine)

- Pipeline: Upload → Parse → Normalize → Rule Engine → Store → Return Results.
- Workers parse PDF/CSV/JSON and compute facts (latest, trend, %change).
- Rule Engine: JSON rules, priority, conditions (AND/OR), explainability.
- Actions: risk alerts, recommendations. Store audit log.
- APIs: Upload report, fetch results, test rules, manage rules.

Database (MongoDB preferred, option: Postgres/NeonDB)

- MongoDB for flexible schemas, Mongoose models.
- Postgres/NeonDB if strong analytics + joins needed.
- Structured Data = CSV, JSON. Semi-structured = JSON/XML. Unstructured = PDF before parsing.
- Canonical JSON schema: patient, source, extracted biomarkers, computed metrics, alerts, recommendations.

Example Rule (JSON)

```
{ "id": "rule-anemia-01", "name": "Anemia detection (adult male)",
"condition": { "all": [ {"fact": "age", "operator":
"greaterThanInclusive", "value": 18}, {"fact": "sex", "operator":
"equal", "value": "male"}, {"fact": "Hemoglobin.latest", "operator":
"lessThan", "value": 13.0} ] }, "action": { "type": "risk", "riskName":
"Anemia suspected", "severity": "moderate", "recommendation": "Repeat CBC
and iron studies" } }
```

18-Hour Work Plan (2 Developers, MERN + Firebase + NeonDB)

Hour	Person A	Person B
0-2	Set up repo, install dependencies (React, Node, Tailwind)	Setup NeonDB / MongoDB + schemas
2-4	Build frontend upload form + file handling	Implement backend file upload API
4-6	Integrate Firebase Auth + JWT in frontend	Worker setup for file parsing jobs
6-8	Create Dashboard (cards, alerts layout)	Implement parser for CSV/JSON → canonical JSON
8-10	Charts (Recharts) + Explainability modal	Implement rule engine evaluator (JSON rules)
10-12	Frontend → API integration (upload, fetch results)	Connect rule evaluation to DB + audit logging

Hour	Person A	Person B
12-14	Admin UI for Rule Editor + test harness	API for rules CRUD + test
14-16	Final frontend polish, progress bars	Integrate recommendations & risk scoring
16-18	QA + end-to-end testing	QA + deploy to test environment