

1 — Project summary & scope (what this system will and will not do)

Purpose (high level)

Provide age- and symptom-based *recommendations* for over-the-counter (OTC) medicines and self-care guidance — and flag serious conditions that require urgent medical attention. The system should **NOT** replace professional medical diagnosis^{**.*} It must include clear disclaimers, triage rules (red flags), and encourage seeing a clinician where appropriate.

Core features (MVP)

- Symptom intake form (age, weight optionally, gender, pregnancy status, allergies, current meds).
- Rule-based triage (red/amber/green).
- Age-aware medicine suggestions for simple, minor complaints (e.g., paracetamol for fever in adult vs child dosing *if allowed by local regulation*).
- Drug interaction & contraindication checks (using authoritative drug vocabularies/APIs).
- Clear safety / warning UI and hard stops for "seek urgent care".
- Audit log of recommendations, versioned clinical rules, and consent capture.

Out of scope (initially)

- Prescription medication issuance.
- Definitive diagnoses.
- Replacing clinician workflows.
- Automated dispensing / medication delivery (unless explicitly required later and vetted legally).

Safety note: This is a high-risk domain. Follow clinical decision support (CDS) guidance and privacy regulations. See the FDA CDS guidance and HIPAA/GDPR sections below.

2 — Regulatory & safety constraints (must-read before design)

1. **Clinical Decision Support regulation (US / FDA):** determine if your recommendation engine falls under FDA definition of a medical device / CDS — design to meet “non-device CDS” criteria or prepare for regulatory requirements.
 2. **Protected health information (HIPAA) if US users or US covered entities involved:** If you store, transmit, or process identifiable health information and you are a covered entity or business associate, HIPAA technical & administrative safeguards apply (encryption at rest/in transit, access controls, breach notifications, auditing).
 3. **EU GDPR — special category data (health):** health data is a “special category” and requires lawful basis + additional safeguards (explicit consent, data minimization, DPIA). If you operate in/serve EU users, GDPR obligations are mandatory.
 4. **Use authoritative drug data & licensing:** For dosing, interactions, contraindications use RxNorm, DrugBank, DailyMed/openFDA or other region appropriate sources. Watch commercial license terms (DrugBank often needs subscription for clinical API).
 5. **User safety & UX:** implement conservative triage (prefer false positives for escalation), always show “see a clinician” for uncertainty, and keep logs/versioning of rules for auditability. See WHO guidance on self-care and published work on chatbot security in health.
-

3 — Data sources & licensing (recommended)

Primary drug vocabularies / clinical data

- **RxNorm** (US normalized drug names, APIs): use for mapping drug names, active ingredients, dose forms.
- **DrugBank Clinical API** (drug interactions, contraindications — commercial terms). ● **openFDA / DailyMed** (US drug labels, adverse events, dosing sections).

Symptom condition / triage knowledge

- Curated clinical rule sets you author or adapt from public guidance (WHO selfcare, NHS symptom checkers when license allows). Avoid using conversational LLMs for final rule decisions without clinician review.

Other

- ICD-10 or SNOMED CT for coding conditions (be mindful of SNOMED license).
- Local medication formularies and pediatric dosing tables (region dependent).

Licensing note: validate license & commercial use (DrugBank may require paid license for clinical module; RxNorm is public). Always record data source and version for each recommendation.

4 — High-level architecture

Diagram (conceptual):

```
+--> ML service (optional) (separate microservice)
+--> Logging / Audit & Monitoring (ELK / Prometheus)
```

Components detail

- **Frontend (Next.js 14+, JS, Tailwind):** symptom pages, age/dose forms, consent modal, results UI, warnings, account pages.
- **Backend (FastAPI):** REST + OpenAPI, auth (OAuth2 / JWT), business logic, rules engine, connectors to drug APIs.
- **DB:** PostgreSQL (encrypted at rest); separate table for anonymized analytics; store only minimal identifiable PHI if required.
- **Cache:** Redis for drug lookups / interaction checks.
- **Rules engine:** deterministic rule set to do triage and age/dose logic. Keep rules versioned in DB or as configuration files.
- **ML (optional):** a classifier (symptoms → probable condition) used only to *suggest* and always surfaced with confidence and rule overrides; final recommendations must be rule-checked.

lua

```
[Next.js Frontend] <--> [FastAPI Backend] <--> [DB (Postgres + Auditing)]
|
+--> Drug data cache (Redis) / RxNorm & DrugBank API calls
+--> Rules engine (Drools / simple Python rule module)
```

- **Audit & Monitoring:** immutable logs for every recommendation, who requested it, rule version, data sources used, timestamp. Use WORM logs or append-only DB schema where regulation requires.
-

5 — Security, privacy & compliance (practical checklist)

Technical safeguards

- TLS 1.2+ everywhere; HSTS.
- Encryption at rest (DB encryption + field-level encryption for PHI).
- Access control: RBAC and least privilege; admin access via MFA.
- JWT for session tokens; rotate keys.
- Rate limiting + WAF (Cloudflare / AWS WAF).
- Secrets management (Vault / AWS Secrets Manager).
- Audit logging & immutable event store for recommendations.

Privacy

- Minimal data collection; anonymize analytics.
- Obtain **explicit consent** for processing health data. Record consent (who, when, for what).
 - Data retention policy & deletion flow (user-initiated deletion).
- Data Processing Agreement (DPA) with cloud providers if you process EU/EAA data.

Organizational

- Appoint Data Protection Officer (GDPR) if required.
- Conduct a DPIA for health data processing.
- Incident response & breach notification procedures (HIPAA: notify within 60 days; follow local law).

(References: HIPAA & GDPR summaries).

6 — Clinical safety design: rules, triage, disclaimers

Triage strategy (recommended conservative approach)

- **Red flags** (immediate referral to emergency): chest pain, severe shortness of breath, loss of consciousness, severe bleeding, suspected stroke signs, severe head injury, very high fever in infants.
- **Amber:** moderate symptoms, vulnerable populations (very young, elderly, pregnant) recommend clinician contact.
- **Green:** self-care / OTC options with dosing instructions and warnings.

Rule engine design

- Rules expressed declaratively (YAML or JSON) with versioning. Example rule fields: `id` , `name` , `conditions` (symptoms, duration, vitals if provided), `age_range` , `pregnancy_flag` , `recommended_actions` , `priority` , `source` (e.g., WHO, local guideline), `created_by` , `version` .
- Each recommendation includes `evidence_source` and `confidence_score` and lists contraindications & interactions evaluated.

Medication checks

- Confirm: age, weight (if pediatric dosing), pregnancy status, allergies, current meds.
- Query drug interaction APIs and contraindication lists. If interaction exists, **do not** recommend and show alternatives. Use a conservative block for serious interactions.

Disclaimer + informed consent

- Prominent banner: “This tool provides information only, not a diagnosis. For emergencies call local emergency services. Always follow your clinician’s advice.” ● Before collecting health data, show consent modal explaining storage, retention, and purpose.

(See FDA CDS guidance and WHO self-care resources).

7 — API specification (core endpoints)

All endpoints return standard problem+json for errors. Use OpenAPI (FastAPI autodoc).

Auth

- `POST /auth/login` returns JWT (access token, refresh token).
- `POST /auth/register` sign up (capture consent).

Symptom intake & triage

- `POST /api/v1/symptoms/assess`
 - Body:

json

```
{
  "user_id": "uuid or null",
  "age_years": 35,
  "age_months": null,
  "weight_kg": 70,
  "sex": "female",
  "pregnant": false,
  "symptoms": ["fever", "headache"],
  "symptom_onset_days": 2,
  "current_medications": ["ibuprofen"],
  "allergies": ["penicillin"],
  "location_country": "IN"
}
```

- Response (abridged):

json

```
{
  "triage": "amber",
  "reasons": ["age > 65", "symptom severity: moderate"],
}
```

```

"recommendations": [
  {
    "id": "rec_001",
    "title": "Paracetamol 500 mg oral",
    "dose": "500 mg every 4–6 hours; max 3 g/day for adults",
    "age_restrictions": ">=12 years",
    "contraindications": ["severe liver disease"],
    "drug_interactions": [],
    "evidence_sources": ["RxNorm vX.Y", "Local formulary v1"]
  }
],
"rule_version": "2025-10-01-v3",
"audit_id": "uuid"
}

```

Drug details

- `GET /api/v1/drugs/{drug_id}` returns drug metadata, interactions, pediatric dosing.

Audit

- `GET /api/v1/audit/{audit_id}` returns logged inputs, rules used, sources.

Admin

- `POST /api/v1/rules` upload/patch rule sets (admin only).
- `GET /api/v1/rules?version=latest` view applied rules.

8 — Database schema (core tables)

Simplified (Postgres):

- `users` (id, email, hashed_password, consent_recorded_at, country, created_at)
- `assessments` (id, user_id nullable, input_json, triage_result, recommendations_json, rule_version, audit_hash, created_at)
- `rules` (id, name, yaml/json, version, author, created_at) `drugs` (rxnorm_id, name, active_ingredient, atc_code, last_synced, source) `drug_interactions` (id, drug_a, drug_b, severity, source) `audit_logs` (id, assessment_id, action, actor, timestamp, signature) Store encrypted `input_json` when it contains identifiers/PHI.

9 — Rules authoring & governance

- Use a clinician-reviewed process: author peer review (2 clinicians) QA test suite (automated scenario tests) publish with semantic versioning.
 - Maintain a **rules rollback** and **changelog**.
 - Each recommendation must link to an evidence source (URL + snapshot of the relevant text) and the rule unit tests that validate it.
-

10 — ML vs Rule-based: recommended hybrid

- **Rules-first (mandatory)**: Triage & medication checks must be deterministic rules.
- **ML (optional assistive layer)**: symptom probable condition ranking (for UX to suggest likely causes). Never auto-apply ML output without rule-based validation. Log model version and confidence. Implement guardrail: if model confidence < threshold, show only rule outputs or escalate.

Model training data: clinical symptom datasets are sensitive; use de-identified public datasets or simulated data; never train on PHI.

11 — Frontend structure & UX patterns (Next.js 14+ + Tailwind)

Pages / flows

- `/` Landing + simple search symptom box.
- `/assess` Symptom intake wizard (stepper): basic demographics symptoms current meds/allergies review & consent results.
- `/results/[audit_id]` Result page with clear triage level, recommendations, what to do next.
- `/account` (optional): saved assessments, share/export, delete data.
- Admin UI (separate app or protected path): rule management, audit viewer.

Key UX elements

- Large, color-coded triage banners (red/amber/green) with accessible text. “Why this recommendation?” modal listing evidence & check results (interactions, contraindications).
- Print / PDF export of assessment with audit id for clinician.
- Explicit “Not a diagnosis” header in results and consent screen.

Accessibility: WCAG AA, large font, ARIA labels for screen readers, keyboard navigation.

12 — Example FastAPI snippet (skeleton)

```
python

# main.py (abridged)
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import uuid, datetime

app = FastAPI(title="SymptomMed API")

class AssessmentIn(BaseModel):
    age_years: int
    symptoms: list[str]
    current_medications: list[str] = []
    allergies: list[str] = []
    pregnant: bool = False
    # ... add fields

@app.post("/api/v1/symptoms/assess")
async def assess(payload: AssessmentIn):
    # 1. basic validation
    if payload.age_years < 0:
        raise HTTPException(400, "Invalid age")
    # 2. run rule engine (placeholder)
    triage = "green" # real logic to replace
    recommendations = [] # call drug rule checks
    audit_id = str(uuid.uuid4())
    # store assessment (save input + result) - omitted
    return {
        "triage": triage,
        "recommendations": recommendations,
        "rule_version": "2025-10-01-v1",
        "audit_id": audit_id
    }
```

I can expand this into a full FastAPI project structure if you want.

13 — Testing & QA

Automated tests

- Unit tests for rule conditions.
- Integration tests: full assessment flows, API DB drug API mocks. End-to-end tests
- (Playwright/Cypress) for important user journeys (consent, assessment, result).
- Security tests (SAST / dependency scanning).
- Penetration test before production if PHI involved.

Clinical validation

- Create a test suite of clinical vignettes (with clinician input) that assert correct triage & meds. Maintain coverage > 95% for core rules.
-

14 — Monitoring, logging & observability

- **Metrics:** assessments/day, triage distribution, API latency, error rates.
 - **Logging:** structured logs with correlation IDs; store audit entries separately. **Alerting:** high error rate, data exfiltration signs, sudden change in triage distribution.
 - **Model / rule drift** track frequency of rule overrides and clinician feedback.
-

15 — Deployment & infra (recommended stack)

- Containerize (Docker).
 - Deploy via managed K8s (EKS/GKE/AKS) or App Platform. Use horizontal autoscaling.
 - PostgreSQL (managed RDS / Cloud SQL) with encryption at rest + daily backups.
 - Redis for caching.
 - Use a secrets manager and IAM roles for service auth.
 - CI/CD: GitHub Actions / GitLab CI — run tests, security scans, then deploy to staging & production with approvals.
 - Use feature flags for new rules (LaunchDarkly / Unleash) to roll out safely.
-

16 — Logging & audit retention policy (example)

- Retain assessment logs for **minimum** 7 years if required by local law (medical records retention varies). Provide user-controlled deletion for non-required jurisdictions. Keep an immutable audit log for compliance, masked where necessary.
-

17 — Team & roles (who you'll need)

- Product owner / clinical lead (licensed clinician).
 - Backend engineer (FastAPI, Postgres, DevOps).
 - Frontend engineer (Next.js 14+, Tailwind).
 - Data engineer (drug APIs, ETL).
 - QA engineer + clinical QA.
 - Security & privacy officer / legal counsel (for HIPAA/GDPR). DevOps / SRE for
 - infra and monitoring.
-

18 — MVP checklist (prioritized)

Phase 1 (MVP)

- Symptom intake + consent flow.
- Rule-based triage & one common condition (e.g., fever/cold) recommendations (adult only).
- RxNorm integration for drug metadata.
- Audit logging & rule versioning.
- Visible disclaimers & emergency red flags. Unit &
- integration tests for rules.

Phase 2

- Pediatric dosing (weight-based).
- Drug interaction checks (DrugBank or openFDA).
- Auth / accounts, export PDF.
- Clinical review pipeline & rule editor.

Phase 3

- Multi-region regulatory compliance, ML assistant (probable diagnostics), BI dashboards, clinician feedback loop.
-

19 — Example risk register (top risks + mitigations)

1. **Incorrect medication suggestion** — mitigate: clinician review, conservative rules, block serious interactions.
2. **Data breach of PHI** — mitigate: encryption, access controls, incident response plan.
3. **Regulatory enforcement (FDA / local)** — mitigate: legal review, avoid features that classify as regulated medical device unless you plan to comply.

20 — Sources & further reading (most-load bearing)

- FDA — Clinical Decision Support Software guidance.
 - HHS — HIPAA Privacy & Security Rule summaries.
 - GDPR — Article 9 (special categories / health data).
 - RxNorm / RxNav APIs — drug normalization & APIs. DrugBank Clinical
 - API (interactions data; commercial). openFDA / DailyMed — public drug
 - label and event data.
 - WHO — self-care for health & minor ailments guidance. Research on AI
 - chatbot security in healthcare.
-

21 — Next steps (practical choices for you right now)

Tell me which of these you want immediately and I'll produce it in this session (I will deliver now, not later):

1. Full **FastAPI** project scaffold (Dockerfile, endpoints, rule engine stub, tests).
2. Full **Next.js 14+** scaffold (pages, Tailwind config, assessment UI components, example integration).
3. A **rules pack** for 10 common symptoms (YAML) + unit tests and clinical vignette suite.
4. A **detailed security & privacy checklist** customized to the countries you'll serve (e.g., India + EU + US).
5. A **sample contract / consent wording** and privacy notice suitable for HIPAA/GDPR audiences.

Pick one (or more) and I'll generate the requested artifact immediately — e.g., a ready-to-run repo scaffold or rules YAML + tests.