Homeopathy Practice Management System

Instructions for AI Coding Assistant (Cursor/Windsurf/etc.)

Project Overview

Build a patient management system for a homeopathy doctor. Offline-first, runs on old laptop, simple and fast.

Tech Stack:

- Frontend: Vite + React + shadcn/ui + Tailwind
- Backend: Flask + SQLAlchemy + SQLite
- No authentication, no cloud, just local

Core Requirements

1. Database Schema

Create 3 tables in SQLite:

patients table:

```sql
```

```
id (primary key, autoincrement)
patient_id (unique, format: P-001, P-002, etc.)
full_name (required, indexed)
date_of_birth (date, required)
age (integer, calculated)
gender (required)
contact_number (required, indexed)
email (optional)
address (text)
occupation (text)
allergies (text, show in red warning)
chronic_conditions (text)
current_medications (text)
family_history (text)
emergency_contact_name (text)
emergency_contact_number (text)
created_at (timestamp)
updated_at (timestamp)
```

visits table:

```sql
id (primary key, autoincrement)
patient_id (foreign key to patients.id, indexed)
visit_date (date, required, indexed)
chief_complaint (text, required)
symptoms (text)
examination_findings (text)
diagnosis (text)
prescription (text - free text, entire prescription)
follow_up_date (date, optional)
doctor_notes (text)
created_at (timestamp)
updated_at (timestamp)
last_edited_at (timestamp, null if never edited)
```

settings table:

```sql
id (primary key, autoincrement)
key (unique, varchar)
value (text)
updated_at (timestamp)
```

Initial settings to insert:

- clinic_name (default: "Gayatri Homeo Clinic")
- clinic_address
- clinic_contact
- clinic_email
- doctor_registration_number
- doctor_qualifications
- letterhead_path (path to uploaded letterhead image for certificates)

---

2. Backend API Endpoints

Base URL: `/api`

Patients:

- `GET /api/patients` - List all (with optional ?search=name&sort_by=name&order=asc)
- `GET /api/patients/:id` - Get one patient + latest visit
- `POST /api/patients` - Create new
- `PUT /api/patients/:id` - Update
- `DELETE /api/patients/:id` - Delete

Visits:

- `GET /api/patients/:patient_id/visits` - All visits for patient (sorted newest first)
- `GET /api/visits/:id` - Get single visit
- `POST /api/patients/:patient_id/visits` - Create new visit
- `PUT /api/visits/:id` - Update visit (set last_edited_at to now)

Analytics:

- `GET /api/analytics/dashboard` - Return:
  - total_patients (count)
  - top_complaints (array of {complaint, count, percentage} - top 3 from last 30 days)
  - age_distribution (object with age groups: 0-18, 19-35, 36-50, 51-65, 65+)

Reports:

- `POST /api/reports/patient/:patient_id` - Generate PDF with all visit history
- `POST /api/reports/certificate` - Generate medical certificate PDF (body: {patient_id, visit_ids[], rest_period, additional_notes})
- `POST /api/reports/prescription/:visit_id` - Generate prescription PDF

Settings:

- `GET /api/settings` - Get all settings as key-value object
- `PUT /api/settings/:key` - Update single setting
- `POST /api/settings/letterhead` - Upload letterhead image (save to static folder, for certificates)

---

3. Frontend Pages & Routes

Header Component (on all pages):

- Logo image (from `frontend/public/logo.png` - hardcoded file path)
- Clinic name "Gayatri Homeo Clinic" next to logo
- Dark mode toggle on the right
- Sticky at top, visible on all pages

Note: Logo file should be placed at `frontend/public/logo.png` before building. No upload functionality needed.

React Router Setup:

```
/ -> HomePage (Dashboard)
/patients -> PatientsPage (list with search)
/patients/new -> AddPatientPage
/patients/:id -> PatientDetailPage
/patients/:id/edit -> EditPatientPage
/patients/:id/visits -> AllVisitsPage
/patients/:id/visits/new -> AddVisitPage
/visits/:id/edit -> EditVisitPage
/settings -> SettingsPage
```

---

4. Page Specifications

HomePage (Dashboard)

Layout:

- 3 stat cards in a row:
    - Total Patients (just number)
    - Top 3 Complaints (last 30 days with %)
    - Age Distribution (simple bar chart)
- Link/button to "View All Patients"

Data: Fetch from `/api/analytics/dashboard`

PatientsPage

Layout:

- Search bar at top (search as you type)
- Sort dropdown (by Name, Last Visit, ID)
- "Add New Patient" button (primary, top-right)
- Vertical list of patient cards

Patient Card shows:

- Patient ID
- Full Name
- Age
- Last Visit Date
- Entire card is clickable -> navigate to /patients/:id

Features:

- Search filters list in real-time (by name, ID, or phone)
- Sorting works without page reload
- Empty state: "No patients yet. Add your first patient!"

PatientDetailPage

Layout (3 sections):

Section 1: Patient Info Card

- All patient details

- Allergies in red/warning box if present
- Edit button -> /patients/:id/edit

Section 2: Latest Visit Card

- Shows most recent visit details
- If visit was edited, show "Last edited: [timestamp]"
- If no visits: "No visits yet"

Section 3: Action Buttons (bottom)

- Back to List
- Edit Patient Info
- View All Visits -> /patients/:id/visits
- Add New Visit -> /patients/:id/visits/new
- Generate Report (dropdown: Visit Report, Certificate, Prescription)

EditPatientPage / AddPatientPage

Form with all patient fields

- Required fields marked with *
- Validation on submit
- Cancel button (go back)
- Save button
- On save: redirect to patient detail page

For AddPatient: Auto-generate patient_id (find max existing, increment)

AllVisitsPage

Layout:

- Patient name/ID at top
- "Add New Visit" button
- List of visit cards (newest first)

Visit Card shows:

- Visit date
- Chief complaint
- Brief symptoms (first 100 chars)
- "Last edited: [time]" if edited
- Click card -> expand to show full details

AddVisitPage / EditVisitPage

Form sections:

1. Visit date (default: today)
2. Chief complaint (textarea)
3. Symptoms (large textarea)
4. Examination findings (textarea)
5. Diagnosis (textarea)
6. Prescription (large textarea - free text)
7. Follow-up date (optional date picker)
8. Doctor's notes (textarea)

Buttons:

- Cancel
- Save Visit
- Save & Print Prescription (saves then opens print dialog with prescription PDF)

For EditVisit: Pre-fill all fields, update last_edited_at on save

---

SettingsPage

Sections:

Clinic Information:

- Form with all clinic settings
- Save button

Letterhead:

- Current letterhead preview (if uploaded) - used in certificates
- Upload new letterhead button (accept PNG/JPG)
- Remove letterhead button

Theme:

- Dark mode toggle switch

---

5. UI Components & Styling

Use shadcn/ui components:

- Button, Card, Input, Textarea, Select, Dialog, Switch, Label

Color Palette (Tailwind config):

Light mode:

- Primary: teal-600
- Background: gray-50
- Text: gray-900
- Danger: red-600

Dark mode:

- Primary: teal-500
- Background: gray-900
- Text: gray-100
- Danger: red-500

Global Rules:

- All buttons min-height: 44px
- All cards: rounded-lg, shadow-sm
- Patient cards: hover effect (scale-[1.02])
- Form inputs: min-height 44px
- Allergies: always show in red bg-red-100 text-red-900 (light) or bg-red-900 text-red-100 (dark)

Dark Mode:

- Toggle in header (top-right corner of every page)
- Use Zustand store to persist preference
- Apply `dark` class to root element

Header Component:

```jsx
// Header.jsx - appears on all pages
// Logo is hardcoded file at /logo.png in public folder
<header className="sticky top-0 z-50 w-full border-b bg-background/95 backdrop-blur">
  <div className="container flex h-16 items-center justify-between">
    <div className="flex items-center gap-3">
      <img src="/logo.png" alt="Logo" className="h-10 w-10" />
      <h1 className="text-xl font-semibold">Gayatri Homeo Clinic</h1>
    </div>
    <ThemeToggle />
  </div>
</header>
```

Place the logo file at `frontend/public/logo.png` before building.

---

6. PDF Generation

For Prescriptions:

```
Header:
  - Clinic name, address, contact (from settings)
  - Doctor qualifications, registration number

Body:
  - Patient name, age, date
  - Prescription text (from visit.prescription)

Footer:
  - Doctor's signature line
```

For Medical Certificates:

```
Header:
  - Letterhead image (if exists) OR text header
  - Clinic details

Body:
  - "This is to certify that..."
  - Patient name, age
  - Visited on: [dates from selected visits]
  - Diagnosis: [from visits]
  - Treatment: [from visits]
  - Advised rest: [rest_period]
  - Additional notes: [additional_notes]

Footer:
  - Date, place
  - Doctor's signature line
```

For Visit Reports:

```
Header: Patient overview

Body: All visits in chronological order
  For each visit:
    - Date
    - Chief complaint
    - Symptoms
    - Prescription
    - Notes

Footer: Summary (total visits, date range)
```

Use @react-pdf/renderer for PDF generation in React

---

7. State Management

Zustand stores:

themeStore.js:

```javascript
```

```
{
  theme: 'light' | 'dark',
  toggleTheme: () => void
}
```

Store in localStorage, apply on mount.

patientStore.js (optional):

```javascript
{
  currentPatient: object | null,
  setCurrentPatient: (patient) => void
}
```

---

8. Project Structure

```
homeopathy-system/
├── frontend/
│   ├── public/
│   │   ├── logo.png (place clinic logo here)
│   │   └── favicon.ico
│   ├── src/
│   │   ├── components/
│   │   │   ├── ui/ (shadcn components)
│   │   │   ├── layout/
│   │   │   │   ├── Header.jsx (logo + clinic name + dark mode toggle)
│   │   │   │   ├── Navbar.jsx
│   │   │   │   └── ThemeToggle.jsx
│   │   │   ├── patients/
│   │   │   │   ├── PatientCard.jsx
│   │   │   │   └── PatientForm.jsx
│   │   │   └── visits/
│   │   │       ├── VisitCard.jsx
│   │   │       └── VisitForm.jsx
│   │   ├── pages/
│   │   │   ├── HomePage.jsx
│   │   │   ├── PatientsPage.jsx
│   │   │   ├── PatientDetailPage.jsx
│   │   │   ├── EditPatientPage.jsx
│   │   │   ├── AddPatientPage.jsx
│   │   │   ├── AllVisitsPage.jsx
│   │   │   ├── AddVisitPage.jsx
│   │   │   ├── EditVisitPage.jsx
│   │   │   └── SettingsPage.jsx
│   │   ├── lib/
│   │   │   ├── api.js (axios instance + all API calls)
│   │   │   └── utils.js
│   │   ├── store/
│   │   │   └── themeStore.js
│   │   ├── App.jsx
│   │   └── main.jsx
│   └── package.json
│
├── backend/
│   ├── app/
│   │   ├── __init__.py (Flask app setup)
│   │   ├── models.py (SQLAlchemy models)
│   │   ├── routes/
│   │   │   ├── patients.py
│   │   │   ├── visits.py
│   │   │   ├── analytics.py
│   │   │   ├── reports.py
│   │   │   └── settings.py
│   │   └── utils/
│   │       └── pdf_generator.py
│   ├── instance/
│   │   └── homeopathy.db
│   ├── static/
│   │   └── letterhead.png
│   ├── requirements.txt
│   └── run.py
```

---

9. Key Implementation Details

Patient ID Generation:

```python
# Find max patient_id, increment
last_patient = Patient.query.order_by(Patient.patient_id.desc()).first()
if last_patient:
    last_num = int(last_patient.patient_id.split('-')[1])
    new_id = f"P-{str(last_num + 1).zfill(3)}"
else:
    new_id = "P-001"
```

Top Complaints Calculation:

```python
# Get visits from last 30 days
thirty_days_ago = datetime.now() - timedelta(days=30)
recent_visits = Visit.query.filter(Visit.visit_date >= thirty_days_ago).all()

# Count complaints
complaint_counts = {}
for visit in recent_visits:
    complaint = visit.chief_complaint
    complaint_counts[complaint] = complaint_counts.get(complaint, 0) + 1

# Get top 3 with percentages
total = len(recent_visits)
top_3 = sorted(complaint_counts.items(), key=lambda x: x[1], reverse=True)[:3]
top_3_with_percent = [
    {
        'complaint': complaint,
        'count': count,
        'percentage': round((count / total) * 100, 1)
    }
    for complaint, count in top_3
]
```

Age Distribution:

```python
def get_age_group(age):
    if age < 19: return '0-18'
    if age < 36: return '19-35'
    if age < 51: return '36-50'
    if age < 66: return '51-65'
    return '65+'

patients = Patient.query.all()
distribution = {'0-18': 0, '19-35': 0, '36-50': 0, '51-65': 0, '65+': 0}
for patient in patients:
    group = get_age_group(patient.age)
    distribution[group] += 1
```

CORS Setup (Flask):

```python
```

```
from flask_cors import CORS

app = Flask(__name__)
CORS(app, resources={r"/api/*": {"origins": "http://localhost:5173"}})
```

API Error Handling Pattern:

```python
@app.route('/api/patients/<int:id>', methods=['GET'])
def get_patient(id):
    try:
        patient = Patient.query.get_or_404(id)
        latest_visit = Visit.query.filter_by(patient_id=id).order_by(Visit.visit_date.desc()).first()
        return jsonify({
            'patient': patient.to_dict(),
            'latest_visit': latest_visit.to_dict() if latest_visit else None
        }), 200
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

Axios Setup (Frontend):

```javascript
```

```
// src/lib/api.js
import axios from 'axios';

const api = axios.create({
  baseURL: 'http://localhost:5000/api',
  headers: { 'Content-Type': 'application/json' }
});

export const getPatients = (search = '', sortBy = 'name', order = 'asc') =>
  api.get('/patients', { params: { search, sort_by: sortBy, order } });

export const getPatient = (id) => api.get(`/patients/${id}`);
export const createPatient = (data) => api.post('/patients', data);
export const updatePatient = (id, data) => api.put(`/patients/${id}`, data);
export const deletePatient = (id) => api.delete(`/patients/${id}`);

export const getPatientVisits = (patientId) => api.get(`/patients/${patientId}/visits`);
export const createVisit = (patientId, data) => api.post(`/patients/${patientId}/visits`, data);
export const updateVisit = (id, data) => api.put(`/visits/${id}`, data);

export const getDashboard = () => api.get('/analytics/dashboard');

export const generatePatientReport = (patientId) =>
  api.post(`/reports/patient/${patientId}`, {}, { responseType: 'blob' });

export const generateCertificate = (data) =>
  api.post('/reports/certificate', data, { responseType: 'blob' });

export default api;
```

---

10. Dependencies

Frontend package.json:

```
json
```

```
{
  "dependencies": {
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-router-dom": "^6.22.0",
    "axios": "^1.6.7",
    "zustand": "^4.5.0",
    "date-fns": "^3.3.1",
    "@react-pdf/renderer": "^3.4.0",
    "lucide-react": "latest"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.2.1",
    "vite": "^5.1.0",
    "tailwindcss": "^3.4.1",
    "autoprefixer": "^10.4.17",
    "postcss": "^8.4.35"
  }
}
```

Backend requirements.txt:

```
flask==3.0.0
flask-cors==4.0.0
flask-sqlalchemy==3.1.1
python-dotenv==1.0.0
reportlab==4.0.9
Pillow==10.2.0
```

---

11. Environment Setup Commands

Frontend:

```bash
npm create vite@latest frontend -- --template react
cd frontend
npm install
npm install react-router-dom axios zustand date-fns @react-pdf/renderer lucide-react
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
npx shadcn@latest init
npx shadcn@latest add button card input textarea select dialog switch label
```

Backend:

```bash
mkdir backend
cd backend
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate
pip install flask flask-cors flask-sqlalchemy python-dotenv reportlab Pillow
pip freeze > requirements.txt
mkdir -p app/routes app/utils instance static
```

---

12. Run Commands

Development:

Terminal 1 (Backend):

```bash
cd backend
source venv/bin/activate
python run.py
# Runs on http://localhost:5000
```

Terminal 2 (Frontend):

```bash
cd frontend
npm run dev
# Runs on http://localhost:5173
```

Production (on uncle's laptop):

```bash
```

```
# Build frontend
cd frontend
npm run build

# Copy to backend
cp -r dist ../backend/static/

# Run Flask (serves both backend + frontend)
cd ../backend
python run.py
# Access at http://localhost:5000
```

---

13. Critical Rules

1. **Header on all pages** - Logo from `/logo.png` (hardcoded) + "Gayatri Homeo Clinic" + dark mode toggle
2. **Logo is static file** - Place at `frontend/public/logo.png`, no upload needed
3. **Never use localStorage/sessionStorage in React** - Use Zustand only for theme
4. **Always calculate age from DOB** - Don't store stale age
5. **Patient ID must be unique** - Check before insert
6. **Visits sorted newest first** - Always
7. **Edit sets last_edited_at** - Only for visits, not patients
8. **No delete for visits** - Only edit allowed
9. **Search is case-insensitive** - Use LIKE/ILIKE in SQL
10. **Allergies always highlighted** - Red background
11. **All dates in ISO format** - YYYY-MM-DD
12. **PDF generation must handle missing letterhead** - Fallback to text
13. **Letterhead is for certificates only** - Logo is separate, for header

---

14. Testing Checklist

Before calling it done:

- [ ] Place logo.png in frontend/public/ folder
- [ ] Header with logo and "Gayatri Homeo Clinic" appears on all pages
- [ ] Letterhead uploads and displays in certificates
- [ ] Add 10 patients with realistic data

- [ ] Add 20 visits across patients
- [ ] Search works with partial names
- [ ] Sort works (all 3 options)
- [ ] Dashboard shows correct stats
- [ ] Edit patient updates correctly
- [ ] Edit visit updates last_edited_at
- [ ] Delete patient removes all visits (cascade)
- [ ] PDF prescription prints correctly
- [ ] Medical certificate includes letterhead
- [ ] Dark mode persists on refresh
- [ ] Settings save and apply
- [ ] All forms validate required fields
- [ ] Allergies show in red
- [ ] Latest visit shows on patient detail
- [ ] Empty states show when no data

Build Order (Recommended)

1. Setup project structure
2. Create database models
3. Create Flask routes (start with patients CRUD)
4. Test API with Postman/Thunder Client
5. Build frontend routing
6. Build PatientsPage (list + search)
7. Build PatientDetailPage
8. Build Add/Edit Patient forms
9. Build visits pages (list, add, edit)
10. Build dashboard with analytics
11. Implement PDF generation
12. Build settings page
13. Add dark mode
14. Polish UI, fix bugs
15. Deploy to uncle's laptop

That's It!

Give this entire document to your AI coding assistant and say:

"Build this app. Start with the database models and Flask routes. Then build the React frontend pages in order. Use the exact tech stack and structure specified."

The AI should be able to build the entire system from these instructions.